SCM Microsystems

Reference Manual – version 2.1

SCL01X

Multiprotocol contactless stationary reader

Reference manual

SCL01X Multiprotocol Contactless Stationary Reader

© SCM Microsystems

Oskar-Messter-Strasse, 13

85737 Ismaning

Germany

Phone +49 89 9595 5000 • Fax +49 89 9595 5555

Document history

Date	Version	Description of change
26/10/2010	2.0	Initial version Typo corrections Add the Pass_through command description Manual fro both SCL010 and SCL011 products
15/03/2011	2.1	Following FW release 1.12 : - Addition of commands for Mifare plus Support (Swtich Card to part 4, Transparent command, Enable Disable polling) - CCID compliance - iCLASS UID support in the Get UID Command.

Contact information

http://www.scmmicro.com/products-services/smart-card-readers-terminals/contactless-dual-interface-readers.html

For sales information, please email sales@scmmicro.com

Table of Contents

1. Legal information 11
   1.1. Disclaimers 11
   1.2. FCC 11
      1.2.1. Section 15.21 Information to user 11
      1.2.2. Section 15, Subpart C, Section 15.225 11
   1.3. Licenses 11
   1.4. Trademarks 12
2. Introduction to the manual 13
   2.1. Objective of the manual 13
   2.2. Target audience 13
   2.3. Product version corresponding to the manual 13
   2.4. Definition of various terms and acronyms 14
   2.5. References 15
   2.6. Conventions 16
3. General information about SCL01X 17
   3.1. SCL01X key benefits 17
   3.2. SCL01X key features 17
   3.3. SCL01X ordering information 18
   3.4. SCL01X customization options 18
   3.5. Contactless communication principles and SCL01X usage recommendations 19
      3.5.1. Power supply 19
      3.5.2. Data exchange 19
      3.5.3. Recommendations 20
   3.6. Applications 21
      3.6.1. General 21
      3.6.2. Applications provided by SCM Microsystems 21
4. SCL01X characteristics 22
   4.1. SCL01X high level architecture 22
      4.1.1. Block diagram 22
      4.1.2. Software architecture 22
   4.2. Quick reference data 23
      4.2.1. SCL01X dimensions 23
      4.2.2. LED behavior 23
      4.2.3. Other data 24
5. Software modules 25
   5.1. Installation 25
   5.2. Utilities 25
   5.3. Driver 25
      5.3.1. SCL01X listing 25
      5.3.2. Supported operating systems 25

5.3.3. PC/SC Part 3 Amendment, Supplemental Document for Contactless ICCs 26
5.3.4. PC/SC 2.0 compliant ATR 26
5.4. Firmware 32
5.4.1. CCID Compliance 32
5.4.2. Automatic PPS 32
6. Commands description 33
6.1. Generic APDU 33
6.1.1. Get UID Command 33
6.1.2. Escape command APDU 35
6.2. Set of APDU for ISO/IEC14443-2 & 3 communications 36
6.2.1. Description 36
6.2.2. Format 36
6.2.3. Response 36
6.2.4. Example 36
6.3. Set of APDU for contactless storage user tokens 37
6.3.1. STORAGE_CARD_CMDS_READ_BINARY 37
6.3.2. STORAGE_CARD_CMDS_WRITE_BINARY 39
6.3.3. STORAGE_CARD_CMDS_LOAD_KEYS 41
6.3.4. STORAGE_CARD_CMDS_AUTHENTICATE 42
6.3.5. STORAGE_CARD_CMDS_VALUE_BLOCK 44
6.4. Set of APDU for ISO/IEC14443-4 user tokens 45
6.4.1. T=CL Command 45
6.4.2. T=CL user command 46
6.5. Switch card to Part 4 47
6.6. Set of APDU defined by SCM Microsystems 47
6.6.1. MIFARE DESFire Commands 47
6.6.2. Commands for communicating with NFC Forum Tags Type 1 47
6.6.3. Commands for communicating with NFC Forum Tags Type 2 56
6.6.4. Commands for communication with NFC Forum Tags Type 3 56
6.6.5. Commands for communicating with NFC Forum Tags Type 4 58
6.6.6. PASS_THROUGH command 59
6.6.7. Set of APDU for ISO/IEC14443-3 user tokens (this needs to be updated as a new section below “Set of APDU for ISO/IEC14443-4 user tokens” 59
6.7. Escape commands 62
6.7.1. Sending escape commands to SCL01X 62
6.7.2. Escape command codes 62
6.7.3. READER_GETCARDINFO 63
6.7.4. READER_LED_CONTROL_BY_FW 63
6.7.5. READER_LEDCONTROL 64
6.7.6. READER_CNTLESS_GET_ATS_ATQB 64
6.7.7. READER_CNTLESS_GET_TYPE 64
6.7.8. READER_CNTLESS_SET_TYPE 65
6.7.9. READER_CNTLESS_RF_SWITCH 66
6.7.10. READER_CNTLESS_RAW_CFG 66
6.7.11. READER_CNTLESS_RAW_XMIT 67
6.7.12. READER_CNTLESS_DISABLE_PPS 68
6.7.13. READER_CNTLESS_848KBPS 68
6.7.14. READER_CNTLESS_BAUDRATE 69
6.7.15. READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2 70
6.7.16. READER_GETPNPPARAMS 71
6.7.17. READER_GETSLOTINFO 71
6.7.18. READER_GET_CARD_DETAILS 72
6.7.19. READER_IS_SCL01x 73
6.7.20. READER_SEND_ATTRIB_WITH_INF 74

6.7.21. USER_TRACEABILITY_DATA 74
6.7.22. READER_DISABLE_ENABLE_POLLING 75
7. Annexes 76
7.1. Annex A – Status words table 76
7.2. Annex B – Sample code using escape commands through Escape IOCTL 77
7.3. Annex C – Mechanical drawings 82
7.3.1. Top casing 82
7.3.2. Bottom casing 83
7.3.3. Cradle 84
7.3.4. Snap-on card holder 85

1. Legal information

1.1. Disclaimers

The content published in this document is believed to be accurate. SCM Microsystems does not, however, provide any representation or warranty regarding the accuracy or completeness of its content and regarding the consequences of the use of information contained herein. If this document has the status “Draft”, its content is still under internal review and yet to be formally validated.

SCM Microsystems reserves the right to change the content of this document without prior notice. The content of this document supersedes the content of previous versions of the same document. The document may contain application descriptions and/or source code examples, which are for illustrative purposes only. SCM Microsystems gives no representation or warranty that such descriptions or examples are suitable for the application that the reader may want to use them for.

Should you notice problems with the provided documentation, please provide your feedback to **support@scmmicro.com.**

1.2. FCC

1.2.1. Section 15.21 Information to user

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment

1.2.2. Section 15, Subpart C, Section 15.225

Note: This equipment has been tested and found to comply with the limits for a Class C digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

--Reorient or relocate the receiving antenna.

--Increase the separation between the equipment and receiver.

--Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

--Consult the dealer or an experienced radio/TV technician for help.

1.3. Licenses

If the document contains source code examples, they are provided for illustrative purposes only and subject to the following restrictions:

- You MAY at your own risk use or modify the source code provided in the document in applications you may develop. You MAY distribute those applications ONLY in form of compiled applications.

- You MAY NOT copy or distribute parts of or the entire source code without prior written consent from SCM Microsystems.
- You MAY NOT combine or distribute the source code provided with Open Source Software or with software developed using Open Source Software in a manner that subjects the source code or any portion thereof to any license obligations of such Open Source Software.

If the document contains technical drawings related to SCM Microsystems products, they are provided for documentation purposes only. SCM Microsystems does not grant you any license to its designs.

1.4. Trademarks

MIFARE is a registered trademark of NXP Semiconductors BV.

FeliCa is a registered trademark of Sony Corporation.

Jewel and Topaz are trademarks of Innovision Research and Technology Plc.

Windows is a trademark of Microsoft Corporation.

2. Introduction to the manual

2.1. Objective of the manual

This manual provides an overview of the hardware and software features of the SCL01X contactless reader, hereafter referred to as “SCL01X”.

This manual describes in details interfaces and supported commands available for developers using SCL01X in their applications.

2.2. Target audience

This document describes the technical implementation of SCL01X.

The manual targets software developers. It assumes knowledge about 13.56 MHz contactless technologies like ISO/IEC 14443 and commonly used engineering terms.

Should you have questions, you may send them to support@scmmicro.com .

2.3. Product version corresponding to the manual

This manual is related to the version indicated below, and all further versions.

Item	Version
Hardware	4.4
Firmware	1.12
Windows Driver	5.09
Linux	2.09
MAC	2.09

2.4. Definition of various terms and acronyms

Term	Expansion
APDU	Application Protocol Data Unit
ATR	Answer to Reset, defined in ISO7816
ATS	Answer to select, defined in ISO/IEC 14443
Byte	Group of 8 bits
CCID	Chip Card Interface Device
CID	Card Identifier
CL	Contactless
DFU	Device Firmware Upgrade
DR	Divider receive: used to determine the baud rate between the reader to the card
DS	Divider send: used to determine the baud rate between the card to the reader
FeliCa™	Sony contactless technology standardized in ISO18092, technology underlying the NFC Forum tag type 3
Jewel/Topaz	Innovision contactless technology, technology underlying the NFC Forum tag type 1
LED	Light emitting diode
MIFARE	The ISO14443 Type A with extensions for security (NXP)
NA	Not applicable
NAD	Node Address
NDEF	NFC Data Exchange Format: data structure defined by the NFC Forum for NFC Forum tags.
NFC	Near Field Communication
Nibble	Group of 4 bits. 1 digit of the hexadecimal representation of a byte. *Example:* 0xA3 is represented in binary as (10100011)b. The least significant nibble is 0x3 or (0011)b and the most significant nibble is 0xA or (1010)b
P2P	Peer - to - Peer
PCD	Proximity Coupling Device
PC/SC	Personal Computer/Smart Card: software interface to communicate between a PC and a smart card
PICC	Proximity Integrated Chip Card
PID	Product ID
Proximity	Distance coverage till ~10 cm.
PUPI	Pseudo unique PICC identifier
RFU	Reserved for future use
RF	Radio Frequency
STCIII	Smart card reader controller ASIC from SCM Microsystems
USB	Universal Serial Bus
VID	Vendor ID
(xyz)b	Binary notation of a number x, y, z $\in \{0,1\}$
0xYY	The byte value YY is represented in hexadecimal

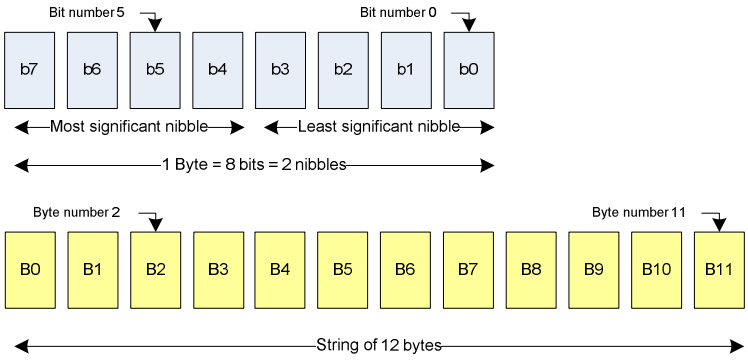
2.5. References

Doc ref in the manual	Description	Issuer
ISO/IEC 7816-4	Identification cards - Integrated circuit(s) cards with contacts Part 4: Interindustry commands for interchange ISO/IEC 7816-4: 1995 (E)	ISO / IEC
ISO/IEC 14443-4	Identification cards — Contactless integrated circuit(s) cards — Proximity cards Part 4: Transmission protocol ISO/IEC 14443-4:2001(E)	ISO / IEC
ISO/IEC 18092	Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1) ISO/IEC 18092:2004(E)	ISO / IEC
NFC Forum tag type 1	NFCForum-TS-Type-1-Tag_1.0	NFC Forum
NFC Forum tag type 2	NFCForum-TS-Type-2-Tag_1.0	NFC Forum
NFC Forum tag type 3	NFCForum-TS-Type-3-Tag_1.0	NFC Forum
NFC Forum tag type 4	NFCForum-TS-Type-4-Tag_1.0	NFC Forum
PC/SC	Interoperability Specification for ICCs and Personal Computer Systems v2.01	PC/SC Workgroup
NFC wrapper	User manual of the NFC wrapper. This manual is part of SCM's Contactless SDK.	SCM Microsystems
CCID	Specification for Integrated Circuit(s) Cards Interface Devices 1.1	USB-IF
USB	Universal Serial Bus Specification 2.0	USB-IF

2.6. Conventions

Bits are represented by lower case ‘b’ where followed by a numbering digit.

Bytes are represented by upper case ‘B’ where followed by a numbering digit.



Example:

163 decimal number is represented

- in hexadecimal as 0xA3
- in binary as (10100011)b

The least significant nibble of 0xA3 is

- 0x3 in hexadecimal
- (0011)b in binary

The most significant nibble of =xA3 is

- 0xA in hexadecimal
- (1010)b in binary

3. General information about SCL01X

3.1. SCL01X key benefits

With its combination of a modern slim design and its state of the art multi-protocol feature set, SCL01X is the perfect desktop reader choice to support various contactless applications such as electronic ID, payment & public transportation schemes and to interact with NFC-enabled devices.

As for all SCM Microsystems products, SCL01X is designed to offer best in class interoperability with various formats of tokens: cards, dongles, watches or NFC mobile phones.

Its infield upgradeable firmware makes SCL01X a secure and future-proof investment providing both flexibility and fast time to market for new applications as well as minimum risk linked to contactless technology standards evolution.

As a latest generation product, SCL01X can be supported by SCM's middleware that resides above the PC/SC API and offers better portability of applications and abstraction of smart card related details that need to be handled by applications developed on top of the PC/SC API.

3.2. SCL01X key features

- Multi-protocol 13.56MHz contactless reader:
  - ISO14443 type A & B,
  - MIFARE,
  - FeliCa™
  - Topaz (NFC Forum tag type 1)
  - NFC Peer-to-peer communication will be available through FW upgrade
- PC/SC v2.0 compliant
- In field upgradeable firmware
- Unique serial number which enables that SCL01X can be plugged into any USB slot on a PC without having to re-install the driver.

3.3. SCL01X ordering information

Item	Part number	
SCL01X	905205	
Cradle	905106	
Snap-on card holder	112302LF	
Contactless SDK	905266	

It is important to note that those accessories are incompatible with each other – i.e. when the snap-on card holder is fitted on the SCL01X, the cradle cannot be used anymore.

3.4. SCL01X customization options

Upon request, SCM can customize:

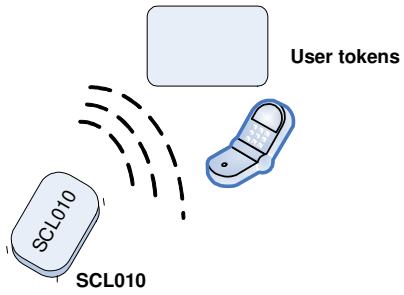
- The color of the casing
- The logo
- The product label
- The USB strings

Terms and conditions apply, please contact your local SCM representative or send an email to sales@scmmicro.com.

3.5. Contactless communication principles and SCL01X usage recommendations

SCL01X is a contactless reader[1] designed to communicate with user tokens.

User tokens[2] are made of a contactless integrated circuit card connected to an antenna



User tokens can take several form factors:

- Credit card sized smart card
- Key fob
- NFC mobile phone etc…

Communication between SCL01X and user tokens uses magnetic field inductive coupling.

The magnetic field is generated by SCL01X has a carrier frequency of 13.56MHz.

3.5.1. Power supply

When the user token is put in the magnetic field of the reader, its antenna couples with the reader and an induction current appears in the antenna thus providing power to the integrated circuit. The generated current is proportional to the magnetic flux going through the antenna of the user token.

3.5.2. Data exchange

The carrier frequency of the magnetic field is used as a fundamental clock signal for the communication between the reader and the card. It is also use as a fundamental clock input for the integrated circuit microprocessor to function.

To send data to the user token the reader modulates the amplitude of the field. There are several amplitude modulation and data encoding rules defined in ISO/IEC 14443 and ISO/IEC 18092. The reader should refer to those standards for further details.

To answer to the reader, the integrated circuit card of the user token modulates its way of loading (impedance) the field generated by the reader. Here also further details can be found in ISO/IEC 14443 and ISO/IEC 18092.

[1] In the ISO/IEC 14443 standard, the reader is called the proximity coupling device (PCD)

[2] In the ISO/IEC 14443 standard, the user token is called proximity integrated chip card (PICC)

3.5.3. Recommendations

The communication between the reader and the user token is sensitive to the presence of material or objects interfering with the magnetic field generated by the reader.

The presence of conductive materials like metal in the vicinity of the reader and the user token can severally degrade the communication and even make it impossible. The magnetic field of the reader generates Eddy or Foucault's currents in the conductive materials; the field is literally absorbed by that kind of material.

It is recommended for proper communication to avoid putting SCL01X in close proximity of conductive materials.

The presence of multiple user tokens in the field also interferes with the communication. When several user tokens are in the field of the reader, load of the field increases which implies that less energy is available for each of them and that the system is detuned. For this reason, SCM Microsystems has implemented in its driver only 1 slot. This means that in the event several user tokens are in the field of the SCL01X, only one will be active.

It is recommended to present only one user credential at a time in front of SCL01X.

The communication between the reader and the user token is sensitive to the geometry of the system {reader, user token}. Parameters like the geometry and specially the relative size of the reader and user token antennas directly influence the inductive coupling and therefore the communication.

SCL01X was primarily designed and optimized to function with user credentials of various technologies having the size of a credit card.

It may happen that SCL01X is not capable of communicating with extremely large or extremely small antennas.

In order to optimize the coupling between the reader and the user token, it is recommended to put both antennas as parallel as possible

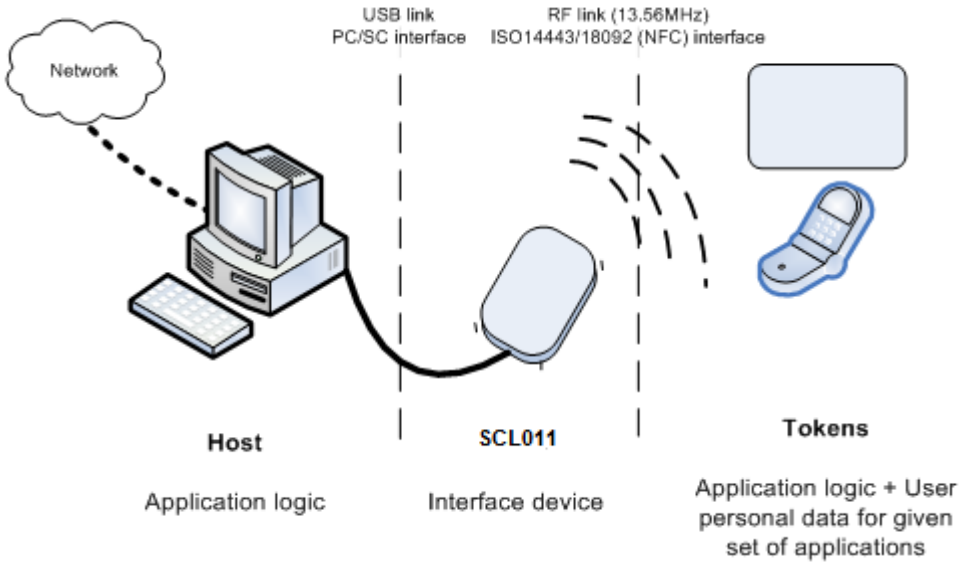
In order to optimize transaction speed between the reader and the card it is recommended to place the user token as close as possible to the reader. This will increase the amount of energy supplied to the user credential which will then be able to use its microprocessor at higher speeds

3.6. Applications

3.6.1. General

SCL01X is a transparent reader designed to interface a personal computer host supporting PC/SC interface with 13.56MHz user tokens like public transport cards, contactless banking cards, NFC forum tags, electronic identification documents – e.g. e-passports, e-ID cards, driving licenses etc.

Those user tokens can have several form factors like credit cards, key fobs, NFC mobile phones or USB dongles like SCT3511 that SCM Microsystems markets.

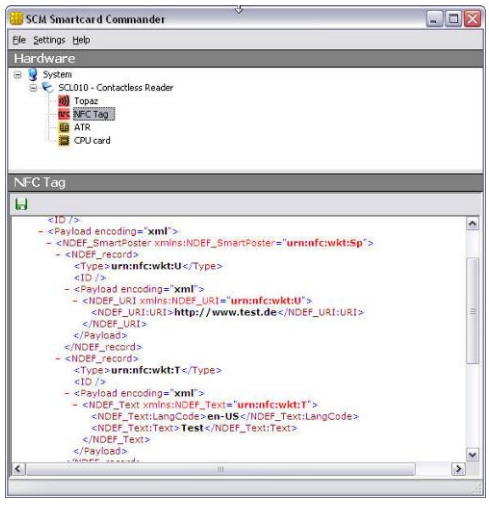


SCL01X itself handles the communication protocol but not the application related to the token. The application-specific logic has to be implemented by software developers on the host.

3.6.2. Applications provided by SCM Microsystems

SCM Microsystems does not provide payment or transport applications.

SCM Microsystems provides a few applications for development and evaluation purposes that can function with SCL01X. They are available within the software development kit. There are many tools provided but the two main ones are:



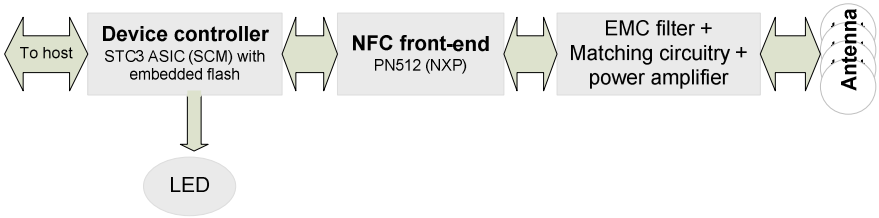
- The NFC forum tag reader/writer is a standalone application that enables the user to read and write NFC forum compliant records into NFC forum compatible tags. It is an easy to use tool to configure rapidly NFC forum tag demonstrations.
- Smart card commander version 1.1 provides NFC forum record parsing functionality of NDEF records in XML format as well as scripting functionality which can be very useful for developers to develop and debug their applications

4.SCL01X characteristics

4.1. SCL01X high level architecture

4.1.1. Block diagram

The link between SCL01X and the host to which it is connected is the USB interface providing both the power and the communication channel.



SCL01X has a device controller which is SCM's STC3 ASIC. This ASIC has several interfaces available. In SCL01X implementation 2 peripherals are connected to the device controller:

- Status indicator LED
- A NFC front-end that handles the RF communication

The ASIC embeds flash memory that is programmed during the manufacturing of SCL01X devices. This flash contains the firmware developed by SCM Microsystems to handle all the RF communication protocols and the PC/SC communication protocol with the host. The flash can be upgraded once the device is deployed on the field, hence enabling firmware upgrades to add and potentially patch features.

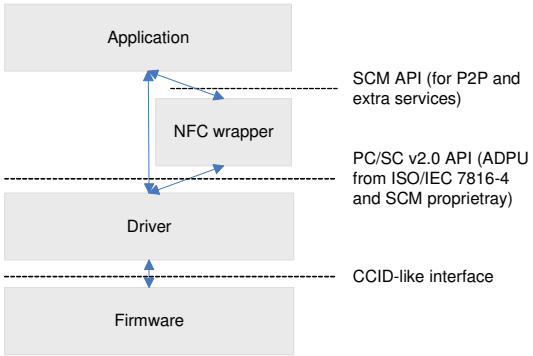
The NFC front-end ensures the coding/decoding/framing modulation/demodulation required for the RF communication. It is controlled by the device controller through registers.

The matching circuitry provides the transmission and receiver paths adaptation for the antenna to function properly.

4.1.2. Software architecture

Applications can interface with the driver directly through the PC/SC interface or through the SCM proprietary interface to the NFC wrapper.

The NFC wrapper simplifies the usage of the different NFC Forum tags with the SCL01X and other SCM contactless readers. It provides a unique API to application developers, which enables them to read and modify NDEF records without further knowledge of the underlying hardware and protocols. Detailed information about the NFC wrapper can be found in SCM's Contactless SDK.

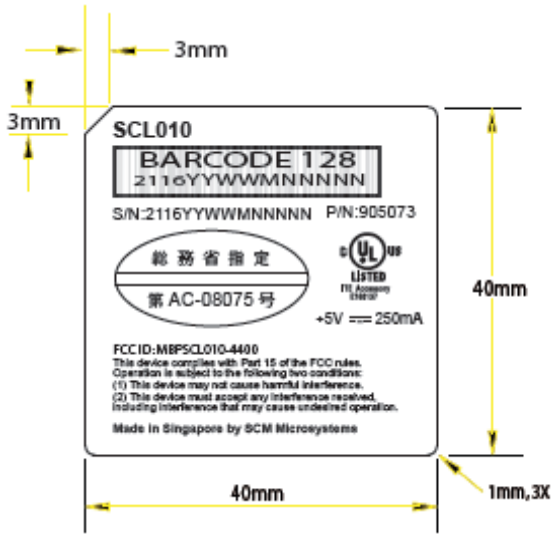


The SCL01X driver implements PC/SC v2.0 API towards upper layers and uses SCM firmware commands encapsulated in CCID-like protocol.

The SCL01X firmware handles all the contactless-related intelligence – i.e. ISO/IEC 14443, ISO/IEC 18092 etc.

4.2. Quick reference data

4.2.1. SCL01X dimensions

Item	Characteristic	Value
SCL01X	Weight	70 Grams
	External dimensions	L 110 mm × W 65mm × H 8mm
	Cable length	1 Meter
	Default color	Cool gray textured
	Default logo	SCM logo
	Default label (SCL010 example)	
Cradle	Weight	32 Grams
	External dimensions	L 83mm × W 80mm × H 40 mm
	Default color	Cool gray
Snap-on card holder	Weight	7 Grams
	External dimensions	L 72mm × W 65mm × H 11.6 mm
	Default color	Translucent black

Drawing with dimensions of the SCL01X and accessories can be found in annex.

4.2.2. LED behavior

The LED behavior of the SCL01X is given below.

SCL01X states	LED Indication (GREEN)
After plug-in (Driver is not installed or driver is installed but either some files related to SCL01X or OS, resource manager is deleted/ missing where there is a situation the reader cannot work	OFF
Just after plug-in (with drivers already installed)	ON
Firmware upgrade running	ON
Just after DFU operation	ON
Suspend / hibernate state	OFF

Reader powered, PICC present in the RF field[3]	500ms ON; 500ms OFF
PICC (token) powered / communication	500ms ON; 500ms OFF
PICC present, but powered down	500ms ON; 500ms OFF
Reader / card errors	100ms ON; 100ms OFF (repeats for 5s)

4.2.3. Other data

Parameter	Value/Description
DC characteristics	High bus powered (SCL01X draws power from USB bus) Voltage: 5V Max. Current : 300mA Suspend current : 230uA
Clock of the device controller	24 MHz
RF carrier frequency	13.56 MHz +/- 50ppm
Modulation	12 to 14 %
USB specification	USB 2.0 FS Devise
USB Speed	Full Speed Device (12Mbit/s)
Device Class	Vendor
PID	0x5291 (SCL010) 0x5293 (SCL01X)
VID	0x04E6
API	PC/SC 2.0
ID1 format tokens supported	NFC forum tag type 1 through SCM-specific APDU NFC forum tag type 2 through PC/SC-defined APDUs NFC forum tag type 3 through SCM-specific APDU NFC forum tag type 4 through PC/SC APDUs ISO/IEC 14443-4 PICC type A and type B MIFARE, Non-Secure FeliCa™ Type B memory card PICC through SCM-proprietary APDU
Maximum baud rate	848 Kbps
Multiple PICC in field	Not supported
Operating temperature range	0º to 50ºC
Operating humidity range	Up to 95%RH non condensing
Certifications	USB, CE, FCC, VCCI, WEEE, RoHS, WHQL, Radio Frequency for Japan,

[3] The LED indicates SCL01X detects a user token in its field but it doesn't guaranty that communication can actually happen

5. Software modules

5.1. Installation

SCM provides an installer for Windows and for MAC

The installer can be used to install the driver as well as some utilities.

5.2. Utilities

The following utilities are available:

- A tool for device firmware upgrade (DFU)
- A tool for testing the installation of the PC/SC driver
- A tool for testing the resource manager
- A tool called *PC/SC Diag* capable of providing basic information about the reader and a card through PC/SC stack

The DFU utility comes with a specific driver for dynamic Device Firmware Upgrade (DFU) through the USB interface.

Operating systems supported by DFU tool:

- Windows 2000
- Windows 2003 Server (32 & 64 bit)
- Windows XP (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Windows Server 2008 (32 & 64 bit)

5.3. Driver

5.3.1. SCL01X listing

SCL01X is listed by PC/SC applications as *SCM Microsystems Inc. SCL01X Contactless Reader*.

5.3.2. Supported operating systems

Operating systems supported by the driver:

- Windows 2000
- Windows 2003 Server (32 & 64 bit)
- Windows XP (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Windows Server 2008 (32 & 64 bit)
- Windows 7 – 32 bits
- Windows 7 - 64 bits

- Mac OS 10.4.11
- Mac OS 10.5.6
- Mac OS 10.6.4
- Debian 5.0 Kernel version 2.6.26 – 32 bit
- Debian 5.0 Kernel version 2.6.26 – 64 bit
- Ubuntu 10.04 Kernel version 2.6.29 – 32 bit
- Ubuntu 10.04 Kernel version 2.6.29 – 64 bit
- Open Suse 11.2 Kernel version 2.6.27 – 32 bit
- Open Suse 11.2 Kernel version 2.6.27 – 64 bit
- Fedora 13 – Kernel version 2.6.33 – 32 bit
- Fedora 13 – Kernel version 2.6.33 – 64 bit

5.3.3. PC/SC Part 3 Amendment, Supplemental Document for Contactless ICCs

The commands specified under PC/SC Part 3 specification amendment "Supplemental Document for Contactless ICCs" should be sent as escape IOCTL to the reader. **ScardTransmit** cannot be used for this purpose.

For the list of commands and their explanation please refer to the PC/SC specification.

First the command is formed as given in the PC/SC specification, command bytes becomes the input buffer for the escape IOCTL.

In order to start a transparent session the command format as given in the PC/SC spec is
"0xFF 0xc2 0x00 0x00 0x02 0x81 0x00 0x00". This has to be sent through **ScardControl** as shown below. The size of the output buffer depends on the commands that we send to the reader hence it is up to the application to allocate the required amount of buffer.

#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE (0xDAC)

lReturn = SCardControl (CardHandle, IOCTL_CCID_ESCAPE, &InBuff, InBufLen, &OutBuff, OutBuffLen, &ResLen);

→ InBuff: "0xFF 0xC2 0x00 0x00 0x02 0x81 0x00 0x00"

← OutBuff: "0xC0 0x03 0x00 0x90 0x00 0x90 0x00".

5.3.4. PC/SC 2.0 compliant ATR

When a user token is placed on the reader, initialization, anti-collision is. The user token is automatically activated and an ATR is built as defined in the PC/SC specification. For NFC Forum tag types 1 and 3, there is no definition in PC/SC.

5.3.4.1. Determining the technology of the user credential

The ScardControl method of PC/SC (see http://msdn.microsoft.com/en-us/library/aa379474(VS.85).aspx) should be used to send the 0x90 IOCTL to SCL3711 in order to determine what type of technology is the user token based on. The output buffer is a BYTE with the following meaning:

Technology	Value
MIFARE1K	0x01
MIFARE4K	0x02
MIFARE Ultralight	0x03
ISO14443-4A	0x04
FeliCa	0x05
Topaz	0x06
ISO14443-4B	0x07

Once a user credential is selected the driver constructs an ATR from the fixed elements that identify the token. Depending on the user technology this ATR can be analyzed as described hereunder.

5.3.4.2. ATR for contactless storage user tokens

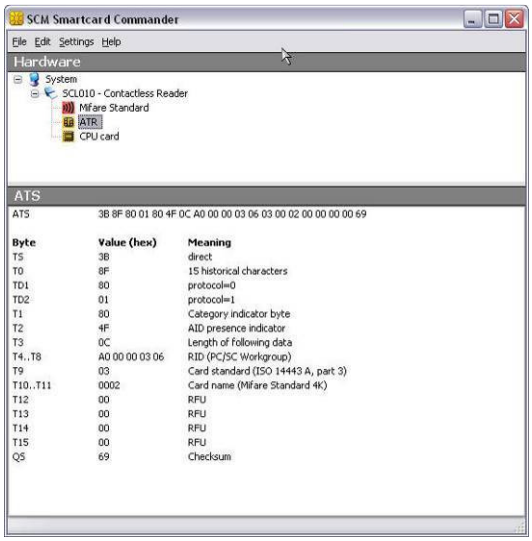
The ATR of the user token is composed as described in the table below. In order to allow the application to identify the storage card properly, it's Standard and Card name describing bytes must be interpreted according to the Part 3 Supplemental Document, maintained by PC/SC.

Tokens using technology like MIFARE are examples of such user tokens.

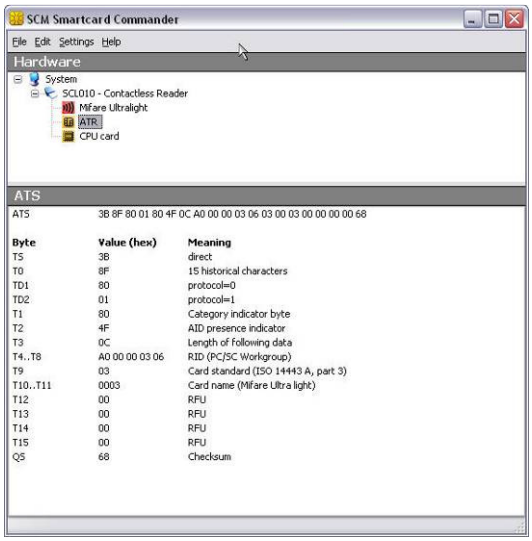
Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	Nibble8 indicates no TA2, TB2, TC2 Nibble 0 means T=0
3	0x01	TD2	Nibble 0 indicates no TA3, TB3, TC3, TD3 Nibble 1 means T=1
4...3+n	0x80		A status indicator may be present in an optional TLV data object
	0x4F	Optional TLV data object	Tag: Application identifier
	Lentgh		1 byte
	RID		Registered identifier on 5 bytes
	PIX		Proprietary identifier extension on 3 bytes
	0x00 0x00 0x00 0x00		4 RFU bytes
4+n		TCK	XOR of all previous bytes

Example of the ATR built for contactless storage tokens:

MIFARE Classic 4K


SCM Smartcard Commander
File Edit Settings Help
Hardware
System
SCL010 - Contactless Reader
Mifare Standard
ATR
CPU card
ATS
ATS 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69
Byte Value (hex) Meaning
TS 3B direct
T0 8F 15 historical characters
TD1 80 protocol=0
TD2 01 protocol=1
T1 80 Category indicator byte
T2 4F AID presence indicator
T3 0C Length of following data
T4..T8 A0 00 00 03 06 RID (PC/SC Workgroup)
T9 03 Card standard (ISO 14443 A, part 3)
T10..T11 0002 Card name (Mifare Standard 4K)
T12 00 RFU
T13 00 RFU
T14 00 RFU
T15 00 RFU
Q5 69 Checksum

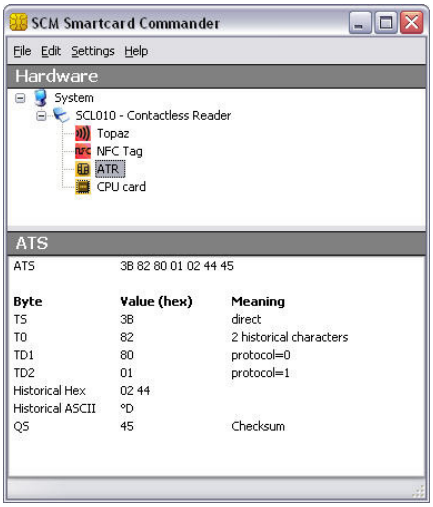
MIFARE Ultralight


SCM Smartcard Commander
File Edit Settings Help
Hardware
System
SCL010 - Contactless Reader
Mifare Ultralight
ATR
CPU card
ATS
ATS 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68
Byte Value (hex) Meaning
TS 3B direct
T0 8F 15 historical characters
TD1 80 protocol=0
TD2 01 protocol=1
T1 80 Category indicator byte
T2 4F AID presence indicator
T3 0C Length of following data
T4..T8 A0 00 00 03 06 RID (PC/SC Workgroup)
T9 03 Card standard (ISO 14443 A, part 3)
T10..T11 0003 Card name (Mifare Ultra light)
T12 00 RFU
T13 00 RFU
T14 00 RFU
T15 00 RFU
Q5 68 Checksum

5.3.4.3. ATR for an NFC Forum tag type 1 user token (Topaz)

Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x82	T0	TD1 present. 2 historical bytes in following ATR
2	0x80	TD1	Nibble 8 indicates no TA2, TB2, TC2 and TD2 present Nibble 0 means T=0
3	0x01	TD2	Nibble 0 indicates no TA3, TB3, TC3 Nibble 1 means T=1
4	0x02	Card Mode	NFC TAG operating at Passive 106 baud rate
5	0x44	Card Type	Card type is Topaz
6	0xXX	TCK	XOR of all previous bytes

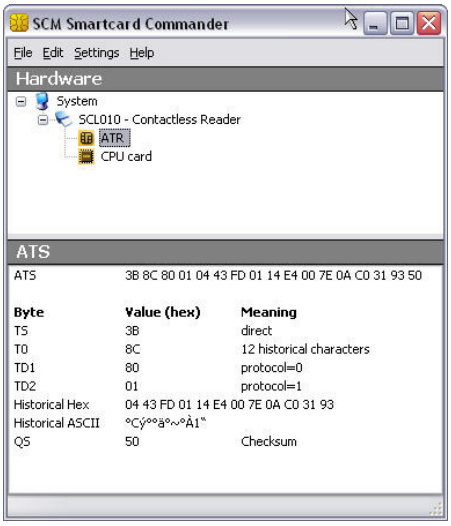
Example of the ATR built for a Topaz tag:



5.3.4.4. ATR for a NFC Forum tag type 3 user token (FeliCa)

Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8C	T0	TD1 present. 12 historical bytes in following ATR
2	0x80	TD1	Nibble8 indicates no TA2, TB2, TC2 and TD2 present Nibble 0 means T=0
3	0x01	TD2	Nibble8 indicates no TA3, TB3, TC3 Nibble 1 means T=1
4	0x04	Card Mode	NFC TAG operating at Passive 212 baud rate
5	0x43	Card Type	Card type is Felica
6	0xFD	IFS	Maximum frame size of felica card
7-14	-	ID	Felica card Identifier – 8 bytes
15	0xXX	Timeout	Write Timeout indicated by card
16	0xXX	TCK	XOR of all previous bytes

Example of the ATR built for a FeliCa user token:


SCM Smartcard Commander
File Edit Settings Help
Hardware
System
SCL010 - Contactless Reader
ATR
CPU card
ATS
ATS 3B 8C 80 01 04 43 FD 01 14 E4 00 7E 0A C0 31 93 50
Byte Value (hex) Meaning
TS 3B direct
T0 8C 12 historical characters
TD1 80 protocol=0
TD2 01 protocol=1
Historical Hex 04 43 FD 01 14 E4 00 7E 0A C0 31 93
Historical ASCII
QS 50 Checksum

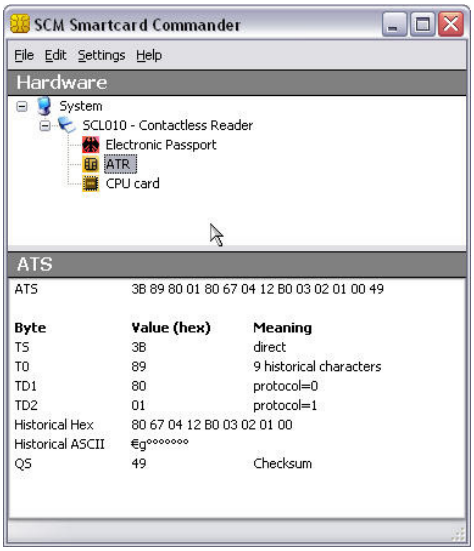
5.3.4.5. ATR for ISO/IEC 14443-4 user tokens

The user token exposes its ATS or application information which is mapped to an ATR. The table describes how this mapping is done.

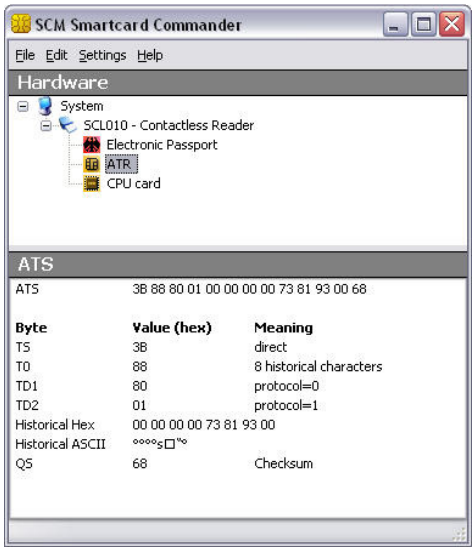
Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	Nibble8 indicates no TA2, TB2, TC2 Nibble 0 means T=0
3	0x01	TD2	Nibble8 indicates no TA3, TB3, TC3 Nibble 1 means T=1
4...3+n		Historical bytes or application information	Type A: the historical bytes from the ATS (up to 15 bytes) Type B (8 bytes): • Byte 0 through 3: application data from ATQB, • Byte 4 through 6: protocol info byte from ATQB, • Byte 7: higest nibble is the MBLI (maximum buffer length index) from ATTRIB, lowest nibble is 0x0
4+n		TCK	XOR of all previous bytes

Example of the ATR built for an ISO14443-4 user tokens:

Type A



Type B



5.4. Firmware

5.4.1. CCID Compliance

The reader firmware is made CCID compliant from firmware Ver 1.11 and above, hence it can be used with Microsoft CCID driver or Linux/MAC open source CCID driver. Windows CCID driver can be got through Windows update and open source CCID driver can be downloaded from the following location "https://alioth.debian.org/frs/?group_id=30105".

In order to make the Linux open source CCID driver load for our device we have to add the VID PID of the device to the "info.plist" file of the driver package.

Escape IOCTL's can be sent as it is with open source CCID driver under Linux/MAC.

5.4.1.1. CCID Error Codes

Extensive error codes are reported on many conditions during all CCID responses. Most of the error messages are reported by the CCID appropriately. Some of the main error codes are:

- HW_ERROR
- XFR_PARITY_ERROR

The following sub-sections discuss when and why these error codes are returned:

5.4.1.1.1. HW_ERROR

This error code is returned when a hardware short circuit condition is detected, during application of power to the card or if any other internal hardware error is detected. This error code has been defined in the error code table 6.2-2 of the CCID specification.

5.4.1.1.2. XFR_PARITY_ERROR

This error code is returned when a parity error condition is detected. This error will be reported in the response to a PC_to_RDR_XfrBlock message. This error code has been defined in the error code table 6.2-2 of the CCID specification.

5.4.2. Automatic PPS

Automatic PPS is implemented in SCL01X's firmware. This means that by default SCL01X switches to the maximum communication speed indicated by the card during its selection. Automatic PPS can be disabled using escape messages as explained later in this manual.

When Auto PPS is disabled (discussed in escape messages section) the reader works at the default baud rate of 106kbps. In PC/SC 2.0 compliant driver an escape command has been introduced to force baud rate.

The maximum speed supported by SCL01X is 848Kbps by default (with 254 bytes frame size). Using escape messages as explained later in this manual it is possible to change this.

6. Commands description

6.1. Generic APDU

6.1.1. Get UID Command

6.1.1.1. Description

GET UID will retrieve the UID or SNR or PUPI of the user token. This command can be used for all supported technologies.

6.1.1.2. I Class support

The Get UID command supports iCLASS serial number.

6.1.1.3. Format

CLA	INS	P1	P2	Lc	Data in	Le
0xFF	0xCA	0x00	0x00	-	-	XX

Setting Le = 0x00 can be used to request the full UID or PUPI is sent back.(e.g. for ISO14443A single 4 bytes, double 7 bytes, triple 10 bytes, for ISO14443B 4 bytes PUPI).

6.1.1.4. Response

Data Out
UID + SW1 + SW2

6.1.1.5. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x62	0x82	End of UID reached before Le bytes (Le is greater than UID length)
0x6C	0xXX	Wrong Length. 0xXX is the exact value for Le

Further error codes can be found in annex

6.1.1.6. Examples

ISO14443-4A	ATR length: 14 ATR: 3B 89 80 01 80 67 04 12 B0 03 02 01 00 49 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 08 24 64 97 (4 byte(s))
ISO14443-4B	ATR length: 13 ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: F0 2C FF FF (4 byte(s))
MIFARE 4K	ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: D4 49 86 7F (4 byte(s))
MIFARE Ultralight	ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 04 E4 C3 D9 5B 02 80 (7 byte(s))
NFC Forum tag type 1	ATR length: 7 ATR: 3B 82 80 01 02 44 45 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 07 83 3C 00 00 02 25 (7 byte(s))
NFC Forum tag type 3	ATR length: 17 ATR: 3B 8C 80 01 04 43 FD 01 14 E4 00 7E 0A C0 31 93 50 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 01 14 E4 00 7E 0A C0 31 (8 byte(s))

6.1.2. Escape command APDU

6.1.2.1. Description

This command can be used to send escape commands to SCL01X. For description of escape commands please refer to the dedicated chapter in this manual.

6.1.2.2. Format

CLA	INS	P1	P2	P3	Data in
0xFF	0xCC	0x00	0x00	Lc	Input buffer of escape command

Lc is the length of the escape command's input buffer. See escape commands description later in this manual

6.1.2.3. Response

Output buffer of the escape command.

6.1.2.4. Example

To get the ATS or ATQB of the ISO14443-4 based user token, you can use this APDU to send the READER_CNTLESS_GET_ATS_ATQB (0x93) escape command

Type A passport

```
ATR length: 14
ATR: 3B 89 80 01 80 67 04 12 B0 03 02 01 00 49
APDU: FF CC 00 00 01 93
SW12: 9000 (OK)
DataOut: 0E 78 33 C4 02 80 67 04 12 B0 03 02 01 00 (14 byte(s))
```

Type B passport

```
ATR length: 13
ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68
APDU: FF CC 00 00 01 93
SW12: 9000 (OK)
DataOut: 50 76 49 FF FF 00 00 00 00 73 81 93 (12 byte(s))
```

To get the reader status about support of 848Kbps, you can use this APDU to send the READER_CNTLESS_848KBPS (0x9D) escape command.

By default the SCL01X doesn't have 848Kbps enabled, the following sequence

- Checks the status (0x00 as response, means 848Kbps is disabled)
- Enables 848Kbps
- Checks the status again and the answer 0x01 indicates 848Kbps is enabled

```
ATR length: 13
ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68
APDU: FF CC 00 00 02 9D FF
SW12: 9000 (OK)
DataOut: 00 (1 byte(s))

APDU: FF CC 00 00 02 9D 01
SW12: 9000 (OK)

APDU: FF CC 00 00 02 9D FF
SW12: 9000 (OK)
DataOut: 01 (1 byte(s))
```

6.2. Set of APDU for ISO/IEC14443-2 & 3 communications

6.2.1. Description

This command can be used to send raw data through the contactless interface. Reader will not add transport protocol data to the raw data. The entire data from the card (including the crc bytes if any) will be returned to the application.

CRC has to be managed manually by the application. It must be added to the raw data by the user. Contactless card's CRC will not be checked by the reader, and will be transmitted in the PICC response data.

Note 1 - Polling

Polling sequence is not stopped automatically when using this command. Between 2 commands the reader will go on checking if the card is still present. This could impact some transaction were no additional sequence is supported in the middle of a transaction. For example, polling must be stopped before starting a secure session (eg. DES authentication) and started after completing the secure session.

Note 2 – CRC

When sending the command, the reader automatically add the CRC. It is not needed to add it in the "raw data" that are sent to the chip.

In the PICC response data, the reader will provide all the bytes that have been sent back by the PICC, including the CRC.

6.2.2. Format

CLA	INS	P1	P2	P3	Data
0xFF	0xEF	0x00	0x00	Length of data	Raw_data

6.2.3. Response

Output buffer
PICC response data

6.2.4. Example

This command can be used to send commands to a MIFARE Ultralight C. The command for generating an 8-byte random number on MIFARE Ultralight C is 0x1A 0x00: Sending the APDU 0xFF 0xEF 0x00 0x00 0x02 0x1A 0x00 will return 0xAF followed by 8 byte random number and the crc bytes. 1. Please note that when ever we initiate a authentication process from the application (e.g AES) we have to disable polling and enable at the end of the session.

6.3. Set of APDU for contactless storage user tokens

6.3.1. STORAGE_CARD_CMDS_READ_BINARY

6.3.1.1. Description

Using this APDU, application can read a memory block on user tokens based on technologies like MIFARE Classic 1K or 4K (block size 0x10 bytes) or MIFARE Ultra light (block size 0x04 bytes).

6.3.1.2. Format

CLA	INS	P1	P2	Le
0xFF	0xB0	RFU	Address	0xXX

Where:

- P2 indicates the block number from where to read
- Le can be a short (maximum value 255) or extended (maximum value 65535). If Le=0x00, then all the bytes until the end of the file are read within the limit of 256 for a short Le field and 65536 for an extended Le field.

6.3.1.3. Response

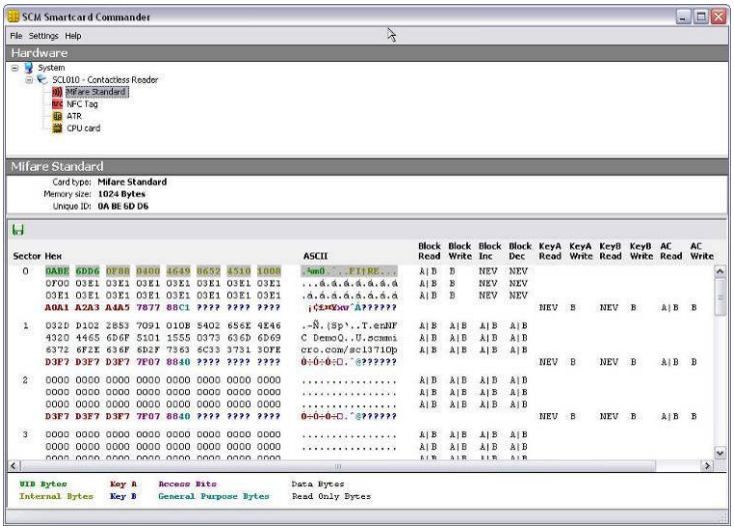
Data Out
Data + SW1 + SW2

6.3.1.4. Status words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x62	0x81	WARNING: part of the returned data may be corrupted
	0x82	WARNING: end of file reached before Le bytes where read
0x67	0x00	Length incorrect
0x68	0x00	CLA byte incorrect
0x69	0x81	Command not supported
	0x82	Security status not satisfied
	0x86	Command not allowed
0x6A	0x81	Function not supported
	0x82	File not found, addressed blocks or bytes do not exist
0x6B	0x00	Wrong P1, P2 parameters
0x6C	0xXX	Wrong Le, 0xXX is the correct value

6.3.1.5. Example

For a MIFARE Classic 1K card which has the following memory content:


SCM Smartcard Commander
File Settings Help
Hardware
System
SCL010 - Contactless Reader
Mifare Standard
NFC Tag
ATR
CPU card
Mifare Standard
Card type: Mifare Standard
Memory size: 1024 Bytes
Unique ID: 0A BE 6D D6
Sector Hex ASCII Block Read Block Write Block Inc Block Dec KeyA Read KeyA Write KeyB Read KeyB Write AC Read AC Write
0 0ABE 6DD6 0F88 0400 4649 8652 4510 1008 .¾mÖ...FIÈRE...
0F00 03E1 03E1 03E1 03E1 03E1 03E1 03E1 ...á.á.á.á.á.á.á
03E1 03E1 03E1 03E1 03E1 03E1 03E1 03E1 .á.á.á.á.á.á.á
A0A1 A2A3 A4A5 7877 88C1 ???? ???? ????
1 032D D102 2853 7091 010B 5402 656E 4E46 .-Ñ.(Sp\..T.enNF
4320 4465 6D6F 5101 1555 0373 636D 6D69 C DemoQ..U.scmmi
6372 6F2E 636F 6D2F 7363 6C33 3731 30FE cro.com/sc13710þ
D3F7 D3F7 D3F7 7F07 8840 ???? ???? ????
2 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
D3F7 D3F7 D3F7 7F07 8840 ???? ???? ????
3 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
MAD Bytes Key A Access Bits Data Bytes
Internal Bytes Key B General Purpose Bytes Read Only Bytes

To read the seventh block, you have to issue the following command and get the following response:

APDU: FF B0 00 06 10
SW12: 9000 (OK)
DataOut: 63 72 6F 2E 63 6F 6D 2F 73 63 6C 33 37 31 30 FE (16 byte(s))

6.3.2. STORAGE_CARD_CMDS_WRITE_BINARY

6.3.2.1. Description

This APDU writes data to a memory address

6.3.2.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0xD6	Address MSB	Address LSB	0xXX	Data

Where:

- P2 indicate the memory block number where data should be written
- Lc=0x10 for MIFARE Classic 1K/4K. Lc=0x04 for MIFARE Ultralight

6.3.2.3. Response

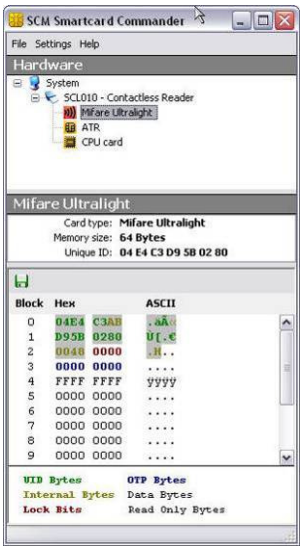
Data Out
SW1 + SW2

6.3.2.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x69	0x81	Command not supported
0x64	0x00	State of the non-volatile memory unchanged

6.3.2.5. Example

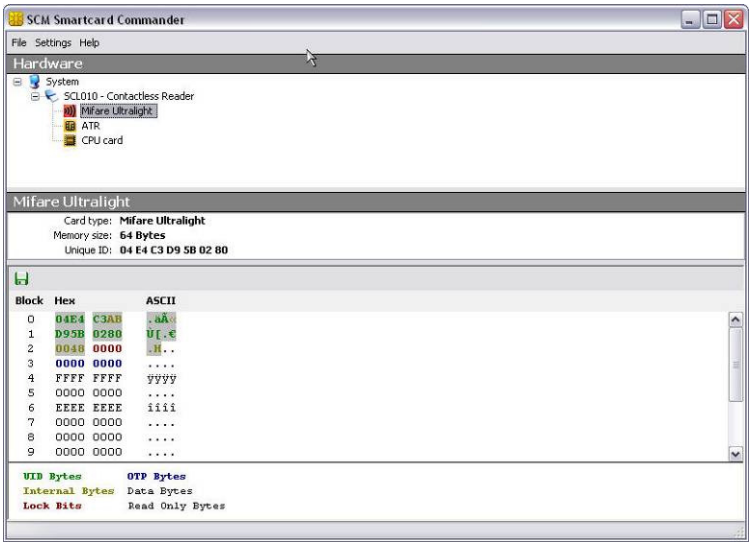
For a MIFARE Classic Ultralight card which has the following memory content:



Issuing the command

ATR length: 20
ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68
APDU: FF D6 00 06 04 EE EE EE EE
SW12: 9000 (OK)

Results into the following memory mapping



6.3.3. STORAGE_CARD_CMDS_LOAD_KEYS

6.3.3.1. Description

Some type of user tokens like MIFARE Classic may require that an authentication happens before any data can be read or written. To perform this authentication, keys need to be loaded in the reader's memory using this command.

6.3.3.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0x82	0x00	Key Type	Key Length	Key value

Where P2 can have the following values (please refer to MIFARE documentation from NXP for further details on what is key A and Key B):

- 0x60 to use the Key A
- 0x61 to use the Key B

6.3.3.3. Response

Data Out
SW1 + SW2

6.3.3.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x69	0x83	Reader key not supported
	0x85	Secured transmission not supported
	0x87	Non volatile memory not available
	0x88	Key number not valid
	0x89	Key length not correct

6.3.4. STORAGE_CARD_CMDS_AUTHENTICATE

6.3.4.1. Description

This command enables to perform authentication for user tokens based on MIFARE Classic 1K or 4K. Before this command can be successfully executed, the STORAGE_CARD_CMDS_LOAD_KEY command must have been executed.

6.3.4.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0x86	0x00	0x00	0x05	Data

Where the data field is structured as follow

Byte #	Value	Description
B0	0x01	Version
B1		Address MSB
B2		Address LSB
B3	0x60	Key A
	0x61	Key B
B4		RFU : should be set to 0x01

Information about memory structure of MIFARE Classic must be requested from NXP Semiconductors.

6.3.4.3. Response

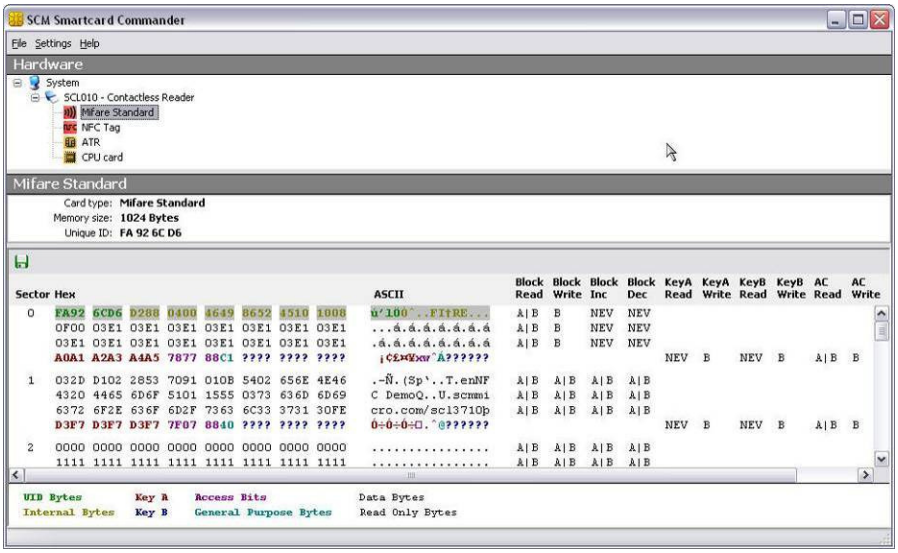
Data Out
SW1 + SW2

6.3.4.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x63	0x00	WARNING no further info
0x69	0x82	Security status not satisfied
	0x84	Referenced key not usable
	0x86	Key type not known

6.3.4.5. Example

For a MIFARE Classic 1K card which has the following memory mapping:



Reading sector 0 or sector 1 of this card requires authentication with key A or key B.

The following example:

- authenticates with key A of sector 0
- reads block #2
- authenticates against sector 1
- reads block #5

```
ATR length: 20
ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
APDU: FF 82 00 60 06 A0 A1 A2 A3 A4 A5
SW12: 9000 (OK)

APDU: FF 86 00 00 05 01 00 02 60 00
SW12: 9000 (OK)

APDU: FF B0 00 02 10
SW12: 9000 (OK)
DataOut: 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 (16 byte(s))

APDU: FF 82 00 60 06 D3 F7 D3 F7 D3 F7
SW12: 9000 (OK)

APDU: FF 86 00 00 05 01 00 05 60 00
SW12: 9000 (OK)

APDU: FF B0 00 05 10
SW12: 9000 (OK)
DataOut: 43 20 44 65 6D 6F 51 01 15 55 03 73 63 6D 6D 69 (16 byte(s))
```

6.3.5. STORAGE_CARD_CMDS_VALUE_BLOCK

6.3.5.1. Description

This APDU is used to interact with MIFARE Classic e-purse applications. Please refer to MIFARE Classic documentation available from NXP Semiconductors for further details on MIFARE classic memory mapping and commands.

6.3.5.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0xF0	0x00	Block#	Lc	Data

Where P2 code the address of the block number addressed

Where the data field is structured as follow

Byte #	Value	Description
B0	0xC0	Increment
	0xC1	Decrement
B1		Block number
B2-B5		Value (LSB first)

6.3.5.3. Response

Data Out
SW1 + SW2

6.3.5.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x67	0x00	Length incorrect
0x68	0x00	CLA byte incorrect
0x6A	0x81	Function not supported
0x6B	0x00	Wrong P1, P2 parameters

6.3.5.5. Example

CLA	INS	P1	P2	Lc	Data in
0xFF	0xF0	0x00	0x1E	0x06	0xC0 0x1E 0x01 0x00 0x00 0x00

Will increment block number 0x1E of a MIFARE Classic-based user token by a value of 0x01.

6.4. Set of APDU for ISO/IEC14443-4 user tokens

6.4.1. T=CL Command

Description

SCL01X can transfer directly ISO/IEC7816-4 APDU to the PICC.

SCL01X supports user tokens that have both the MIFARE and T=CL partitions. Depending on the APDU sent by the host, the reader switches to the corresponding mode (MIFARE or T=CL) automatically and the command is processed accordingly.

6.4.1.1. Format

CLA	INS	P1	P2	P3	Data

Description of the APDU commands can be found in ISO/IEC 7816-4 specification.

6.4.1.2. Response

Data Out
PICC answer as defined in ISO/IEC 7816-4+ SW1 + SW2

As defined in ISO/IEC 7816-4.

6.4.1.3. Status Words

SW1	SW2	Description
		See ISO/IEC 7816-4

As defined in ISO/IEC 7816-4.

6.4.1.4. Example

The following APDU sequence reads the first 256 bytes of the data group 1 as specified in ICAO LDS (logical data structure) for machine readable travel documents with open access. It first selects the issuer application using its AID (0xA0 0x00 0x00 0x02 0x47 0x10 0x01), then selects the DG1 file (0x01 0x01) and then does a read binary.

```
APDU: 00 A4 04 0C 07 A0 00 00 02 47 10 01
SW12: 9000 (OK)

APDU: 00 A4 02 0C 02 01 01
SW12: 9000 (OK)

APDU: 00 B0 00 00 00
SW12: 9000 (OK)
DataOut: 61 5B 5F 1F 58 50 3C 55 54 4F 45 52 49 4B 53 4F 4E 3C 3C 41 4E 4E 41 3C 4D 41 52 49 41 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 4C 38 39 38 39 30 32 43 3C 3
```

6.4.2. T=CL user command

Description

This command can be used to send raw data to the user token.

6.4.2.1. Format

CLA	INS	P1	P2	P3	Data
0xFF	0xFE	0x00	0x00	Lraw_data	Raw_data

6.4.2.2. Response

Data Out
PICC response data+ SW1 + SW2

6.4.2.3. Status Words

SW1	SW2	Description

User should refer to the status words defined by the PICC manufacturer for a description of the status words

6.4.2.4. Example

Let's consider the Select command defined in ISO7816-4. This command being ISO can be sent to the user token in 2 different ways:

- Using the T=CL command
- Using the T=CL user command

Here are the 2 answers for the select command:

```
ATR length: 19
ATR: 3B 89 80 01 4D 54 43 4F 53 73 01 01 01 3C
APDU: 00 A4 00 00
SW12: 9000 (OK)

APDU: FF FE 00 00 04 00 A4 00 00
SW12: 9000 (OK)
```

The T=CL command is nevertheless more useful for sending commands which are not defined in ISO7816.

6.5. Switch card to Part 4

This user command is mainly targeted at Mifare plus S0 cards. Mifare plus card at S0 level gets detected as Mifare memory card. In order to personalize these cards first it needs to be switched to Part 4 mode. For this purpose this user command needs to be issued using ScardTransmit function. Command sample and response is given below.

Format

CLA	INS	P1	P2	P3
0xFF	0xF8	0x00	0x00	0x00

Response

Output buffer
0x90 0x00

6.6. Set of APDU defined by SCM Microsystems

6.6.1. MIFARE DESFire Commands

Description

This command can be used to send commands to DESFire-based user tokens.

For a description of DESFire commands please contact NXP Semiconductors.

6.6.1.1. Format

CLA	INS	P1	P2	P3	Data
0xFF	0xDE	0x00	0x00	Lcommand	Command

Response

Data Out
DESFire response data+ SW1 + SW2

6.6.2. Commands for communicating with NFC Forum Tags Type 1

Commands for Static and Dynamic Memory Models

- Read Identification (RID)
- Read All Blocks 0 – Eh (RALL)
- Read Byte (READ)
- Write-Erase Byte (WRITE-E)
- Write-No-Erase Byte (WRITE-NE)

Commands for Dynamic Memory Model

- Read Segment (RSEG)
- Read 8 Bytes (READ8)
- Write-Erase 8 Bytes (WRITE-E8)
- Write-No-Erase 8 Bytes (WRITE-NE8)

6.6.2.1. Read Identification (RID)

Description

This command is used to retrieve the tag's identification.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x50	0x00	0x00	0x00	-

Response

Data	SW1 SW2
HR0 HR1 UID0 UID1 UID2 UID3	0x90 0x00

Where

- HR0 and HR1 are the 2 bytes Header ROM which identify the tag
- UID0 through UID3 are the first 3 bytes of the tag's UID.

Topaz tags have a 7 bytes long UID which can be fully fetched using the Get UID APDU described earlier in this manual.

6.6.2.2. Read All Blocks (RALL)

Description

The RALL command reads-out the two header ROM bytes and the whole of the static memory blocks 0x0-0xE.

Format

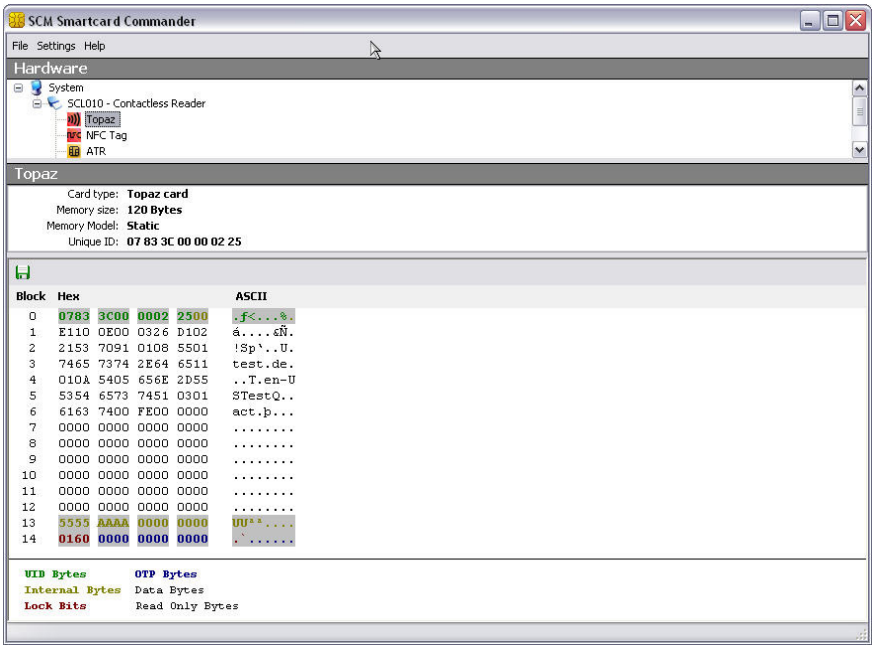
CLA	INS	P1	P2	P3	Data
0xFF	0x52	0x00	0x00	0x00	-

Response

Data	SW1 SW2
HR0 HR1 120 bytes (Blocks 0 – E)	0x90 0x00

Example

For an NFC Forum tag type 1 that has the following memory content



The following 2 APDUs can be sent to retrieve the UID and read all the memory blocks

```
ATR length: 7
ATR: 3B 82 80 01 02 44 45
APDU: FF 50 00 00 00
SW12: 9000 (OK)
DataOut: 11 48 07 83 3C 00 (6 byte(s))

APDU: FF 52 00 00 00
SW12: 9000 (OK)
DataOut: 11 48 07 83 3C 00 00 02 25 00 E1 10 0E 00 03 26 D1 02 21 53 70 91 01 08 55 01 74 65 73 74 2E 64 65 11 01 0A 54 05 65 6E 2D 55 53 54 65 73 74 51 03 01 61 63 74
```

6.6.2.3. Read Byte (READ)

Description

This command reads a single EEPROM memory byte within the static memory model area of blocks 0x0-0xE.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x54	0x00	Byte Address	0x00	-

Where P2 codes the address of the memory byte in the following way:

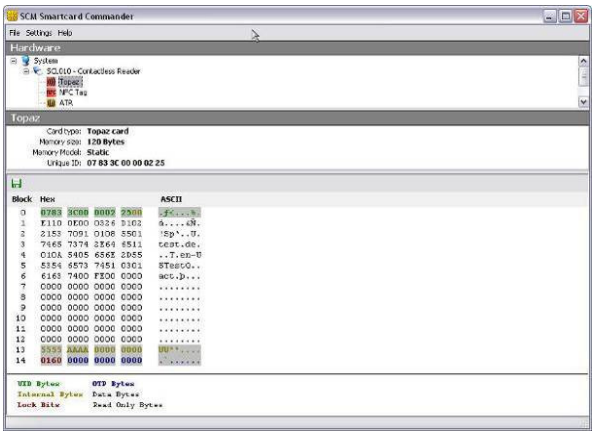
Bit numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response

Data	SW1 SW2
1 byte of data	9000

Example

For an NFC forum tag type 1 which has the following memory dump



Sending the following sequence of APDU will read byte 0x01 of block 0x00 through 0x04

```
ATR length: 7
ATR: 3B 82 80 01 02 44 45
APDU: FF 54 00 01 00
SW12: 9000 (OK)
DataOut: 83 (1 byte(s))

APDU: FF 54 00 09 00
SW12: 9000 (OK)
DataOut: 10 (1 byte(s))

APDU: FF 54 00 11 00
SW12: 9000 (OK)
DataOut: 53 (1 byte(s))

APDU: FF 54 00 19 00
SW12: 9000 (OK)
DataOut: 65 (1 byte(s))

APDU: FF 54 00 21 00
SW12: 9000 (OK)
DataOut: 0A (1 byte(s))
```

6.6.2.4. Write-Erase Byte (WRITE-E)

Description

This commands erases and then writes the value of an individual memory byte within the static memory model area of blocks 0x0-0xE.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x56	0x00	Byte Address	0x01	1 byte of data to be written

Where P2 codes the address of the memory byte in the following way:

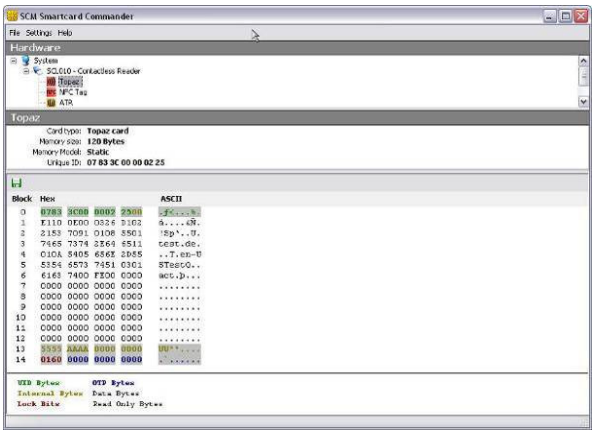
Bit numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response

Data	SW1 SW2
Byte value that has been written	0x90 0x00

Example

For an NFC forum tag type 1 which has the following memory dump



The following sequence does

- A READ of the byte # 0x03 in block 0x07
- A WRITE ERASE of the byte # 0x03 in the block =0x07 with value 0x0A
- A READ of the byte # 0x03 in block 0x07
- A WRITE ERASE of the byte # 0x03 in the block =0x07 with value 0x02
- A READ of the byte # 0x03 in block 0x07

```
ATR length: 7
ATR: 3B 82 80 01 02 44 45
APDU: FF 54 00 3B 00
SW12: 9000 (OK)
DataOut: 00 (1 byte(s))

APDU: FF 56 00 3B 01 0A
SW12: 9000 (OK)
DataOut: 0A (1 byte(s))

APDU: FF 54 00 3B 00
SW12: 9000 (OK)
DataOut: 0A (1 byte(s))

APDU: FF 56 00 3B 01 02
SW12: 9000 (OK)
DataOut: 02 (1 byte(s))

APDU: FF 54 00 3B 00
SW12: 9000 (OK)
DataOut: 02 (1 byte(s))
```

6.6.2.5. Write-No-Erase Byte (WRITE-NE)

Description

This command writes a byte value on an individual memory byte within the static memory model area of blocks 0x0-0xE. This command does not erase the value of the targeted byte before writing the new data. Execution time of this command by NFC Forum tags type 1, is approximately half that of the normal write command (WRITE-E). Using this command, EEPROM bits can be set but not reset.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x58	0x00	Byte Address	0x01	1 byte of data to be written

Where P2 codes the address of the memory byte in the following way:

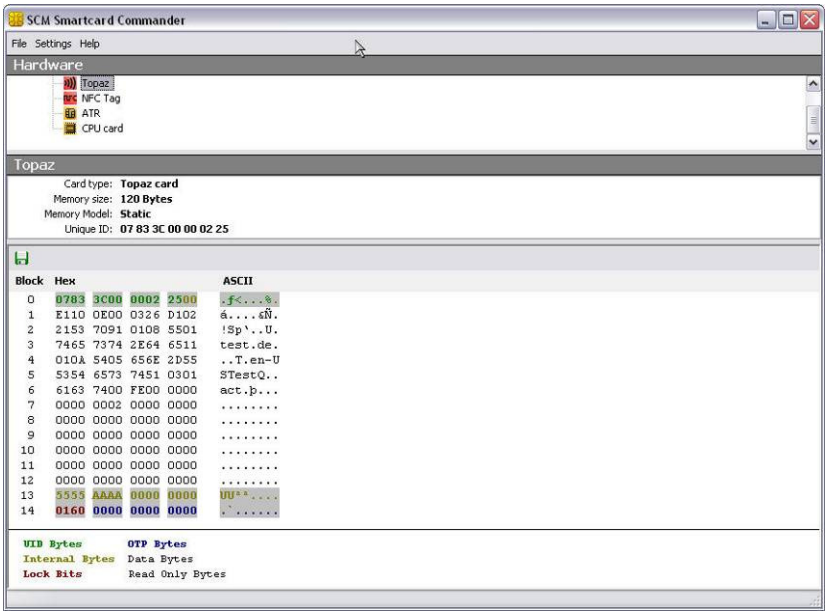
Bit numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response

Data	SW1 SW2
Value of the memory byte after execution	0x90 0x00

Example

For an NFC forum tag type 1 which has the following memory dump



The following sequence does

- A READ of the byte # 0x03 in block 0x07
- A WRITE NO ERASE of the byte # 0x03 in the block =0x07 with value 0x09
- A READ of the byte # 0x03 in block 0x07

```
ATR length: 7
ATR: 3B 82 80 01 02 44 45
APDU: FF 54 00 3B 00
SW12: 9000 (OK)
DataOut: 02 (1 byte(s))

APDU: FF 58 00 3B 01 09
SW12: 9000 (OK)
DataOut: 0B (1 byte(s))

APDU: FF 54 00 3B 00
SW12: 9000 (OK)
DataOut: 0B (1 byte(s))
```

One can notice that the value of the byte after the WRITE NO ERASE is not 0x09 but 0x0B because the memory value before the operation was 0x02. The memory result of a WRITE NO ERASE command is the logical OR of the value before the operation and the value written:

description	Value (binary)	Value (0x)
Memory value before	(00000010)b	0x02
Value written	(00001001)b	0x09
Result	(00001011)b	0x0B

6.6.2.6. Read Segment (RSEG)

Description

This command reads out a complete segment (or block) of the memory a Topaz tag with dynamic memory model. Please note that this command only function on those specific Topaz tags.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x5A	0x00	Segment Address	0x00	-

P2 – Segment Address - b8 - b5 - Segment (0x0 – 0xF)

b4 - b1 - 0

Response

Data	SW1 SW2
128 bytes of data	0x90 0x00

6.6.2.7. Read 8 bytes (READ8)

Description

This command reads out a block of memory.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x5C	0x00	Block Address	0x00	-

P2 – Block Address - b8 - b1 - General block (0x00 -0xFF)

Response

Data	SW1 SW2
8 bytes of data	0x90 0x00

6.6.2.8. Write-Erase 8 bytes (WRITE-E8)

Description

This command writes erases a memory block and then writes a value to it.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x5E	0x00	Block Address	0x08	8 bytes of data to be written

P2 – Block Address - b8 - b1 - General block (0x00 – 0xFF)

Response

Data	SW1 SW2
8 bytes of data that have been written	0x90 0x00

6.6.2.9. Write-No-Erase 8 bytes (WRITE-NE8)

Description

The WRITE-E8 command writes with no erase to a block of memory.

This command does not erase the value of the targeted block before writing the new data. Using this command, EEPROM bits can be set but not reset.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x60	0x00	Block Address	0x08	8 bytes of data to be written

P2 – Block Address - b8 - b1 - General block (0x00 – 0xFF)

Response

Data	SW1 SW2
8 bytes of data	0x90 0x00

Example

Sending the following command to an NFC Forum type 1 tag that has the value (0x01 0x02 0x03 0x04 0x00 0x00 0x00 0x00) in the first EEPROM block

CLA	INS	P1	P2	P3	Data
0xFF	0x60	0x00	0x00	0x08	0x00 0x01 0x03 0x04 0x05 0x06 0x07 0x08

Will give the answer

Data	SW1 SW2
0x01 0x03 0x03 0x04 0x05 0x06 0x07 0x08	0x90 0x00

6.6.3. Commands for communicating with NFC Forum Tags Type 2

To interact with an NFC Forum tag type 2, the commands defined for memory storage cards are to be used (STORAGE_CMDS_READ_BINARY and STORAGE_CMDS_WRITE_BINARY with block length of 0x04.

Please refer to *NFC Forum tag type 2* specification for definition of the commands to be used

6.6.4. Commands for communication with NFC Forum Tags Type 3

Proprietary APDUs defined for the following FeliCa™ non-secure commands are described in this section. SCL01X does not support FeliCa™ secure commands.

- REQC
- Request Service
- Request Response
- Read
- Write

6.6.4.1. REQC

Description

This command is used to detect the presence of a NFC Forum tag type 3 in the field

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x40	0x00	0x00	0x04	2 bytes of system code, 1 byte RFU, 1 byte TSN

Response

Data	SW1 SW2
16 bytes of NFCID2 + 2 bytes of System Code (sent only if the RFU byte is 0x01)	0x90 0x00

6.6.4.2. Request Service

Description

On receiving this command an NFC Forum tag 3 type will respond with the area key version of the specified area and the service key version of the specified service.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x42	Number of services/areas	0x00	2 * P1	Service Code List / Area Code List

Response

Data	SW1 SW2
8 bytes IDm + No. of Service or areas(n) + Service version or area version list (2*n)	0x90 0x00

6.6.4.3. Request response

Description

When an NFC Forum tag type 3 receives this command, it responds with its current mode (0/1/2).

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x44	0x00	0x00	0x00	-

Response

Data	SW1 SW2
8 bytes IDm + Mode	0x90 0x00

6.6.4.4. Read

Description

When an NFC Forum tag type 3 receives this command, it responds with the record value of the specified service.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x46	Number of services	Number of blocks	2*(P1 + P2)	Service Code List, Block List

Response

Data	SW1 SW2
8 bytes IDm + Status Flag 1 + Status Flag 2 + No. of blocks(n) + Block data (n*16)	0x90 0x00

6.6.4.5. Write

Description

When an NFC Forum tag type 3 receives this command, it writes the records of the specified service.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x48	Number of services	Number of blocks	2*(P1 + P2) + (16 * P2)	Service Code List, Block List, Block Data

Response

Data	SW1 SW2
8 bytes IDm + Status Flag 1 + Status Flag 2	0x90 0x00

6.6.4.6. Request System Code

Description

<TBC>

Format

CLA	INS	P1	P2	P3	Data
0xFF	0x4A	0x00	0x00	0x00	-

Response

Data	SW1 SW2
8 bytes IDm + No. of System Codes (n) + System Code List (2n)	0x90 0x00

6.6.5. Commands for communicating with NFC Forum Tags Type 4

There is no need to define specific APDU commands as command T=CL command already defined earlier can be used.

Please refer to *NFC Forum tag type 4* specification for definition of the commands to be used

6.6.6. PASS_THROUGH command

Description

This command can be used to send raw data to the user token. SCL3711 will not add transport protocol data to the raw data – e.g. PCB, NAD, CID etc.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0xEF	0x00	0x00	Lraw_data	Raw_data

Response

Data
PICC response data

Example

This command can be used to send commands to a MIFARE Ultralight C

The command for generating an 8-byte random number on MIFARE Ultralight C is 0x1A 0x00:

Sending the APDU 0xFF 0xEF 0x00 0x00 0x02 0x1A 0x00

Will return 0xAF followed by 8 byte random number

6.6.7. Set of APDU for ISO/IEC14443-3 user tokens (this needs to be updated as a new section below "Set of APDU for ISO/IEC14443-4 user tokens"

Description

This command can be used to send raw data to the user token. Reader will not add transport protocol data to the raw data. The entire data from the card (including the crc bytes if any) will be returned to the application. Polling must be stopped before starting a secure session (eg. DES authentication) and started after completing the secure session.

Format

CLA	INS	P1	P2	P3	Data
0xFF	0xEF	0x00	0x00	Length of data	Raw_data

Response

Output buffer
PICC response data

Example

This command can be used to send commands to a MIFARE Ultralight C.
The command for generating an 8-byte random number on MIFARE Ultralight C is 0x1A 0x00:
Sending the APDU 0xFF 0xEF 0x00 0x00 0x02 0x1A 0x00 will return 0xAF followed by 8 byte

random number and the crc bytes. 1. Please note that when ever we initiate a authentication process from the application (e.g AES) we have to disable polling and enable at the end of session.

Special notes on Mifare Plus cards support

When a Mifare plus card is in S0 mode in order to customize the card the following steps needs to be done.

1. First it has to be moved to part 4 mode using the command given under “User command to manually switch card to Part 4” section.
2. Then we need to stop the polling of the card using the escape IOCTL “READER_DISABLE_ENABLE_POLLING”.
3. Please note that when ever we initiate a authentication process from the application (e.g AES) we have to disable polling and enable at the end of session.
4. Then we have to use “T=CL user command” given under section 6.3.2 to initiate and complete the authentication process. This user command can be used to send all proprietary commands defined in the card specification.
5. At the end of communication with the card we need to start the polling again using “READER_DISABLE_ENABLE_POLLING” IOCTL. **This is very important else card tracking will not work.**

6.7. Escape commands

6.7.1. Sending escape commands to SCL01X

A developer can use 2 methods to send escape commands to SCL01X

- SCardControl method defined in PC/SC API
- SCardTransmit method defined in PC/SC API in conjunction with the escape command APDU defined earlier in this manual

6.7.2. Escape command codes

Escape commands can be used by an application to configure SCL01X to function in a mode that is not its default configured mode or to get specific information. To put the SCL01X back into its default mode, either the SCL01X has to be unplugged and plugged again or the application can send again the same escape command.

The following escape commands are supported by SCL01X.

Escape command	Code
READER_GETCARDINFO	0x11
READER_LEDCONTROL	0x19
READER_CNTLESS_GET_ATS_ATQB	0x93
READER_CNTLESS_GET_TYPE	0x94
READER_CNTLESS_SET_TYPE	0x95
READER_CNTLESS_RF_SWITCH	0x96
READER_CNTLESS_RAW_CFG	0x97
READER_CNTLESS_RAW_XMIT	0x98
READER_ CNTLESS_DISABLE_PPS	0x99
READER_CNTLESS_848KBPS	0x9D
READER_CNTLESS_BAUDRATE	0x9E
READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2	0xAD
READER_LED_CONTROL_BY_FW	0xB2
READER_GETPNPPARAMS	0xD3
READER_GETSLOTINFO	0xD9
READER_GET_CARD_DETAILS	0xDA
READER_IS_SCL01X	0xDB
READER_SEND_ATTRIB_WITH_INF	0xE2
READER_GET_CARD_TYPE	0xE3
READER_IS_COLLISION_DETECTED	0xE4
FELICA_PASS_THROUGH	0xF3
USER_TRACEABILITY_AREA	0xF0
READER_DISABLE_ENABLE_POLLING	0xAC

Sample code to send escape commands can be found in annex.

6.7.3. READER_GETCARDINFO

This escape command is used to get information about the card placed on the reader. The SCL01X returns an error if no card is placed on it.

The input buffer shall contain the escape command code

Input buffer
0x11

The output buffer contents are described below.

Output buffer	Value	Description
B0	0x01	Contactless card present
B1	0xNN	Baud rate of card-reader communication (see READER_CNTLESS_BAUDRATE escape command)
B2 – Upper nibble	0x0_	Memory Card
	0x1_	T = CL card
	0x2_	Dual Mode Card
B2 – Lower nibble	0x_0	Type A card + Topaz
	0x_1	Type B card
	0x_4	Felica (212 kbps card)
	0x_8	Felica (424 kbps card)

6.7.4. READER_LED_CONTROL_BY_FW

This escape command may be used to enable or disable LED control by the firmware.

The input buffer is

Byte #	Value	Description
B0	0xB2	Escape command code
B1	0x00	Disable LED control by FW
	0x01	Enable LED control by FW

The output buffer is

Output buffer
NULL

6.7.5. READER_LEDCONTROL

This escape command is used to turn ON/OFF the LED.

This escape command shall work only if LED control by firmware is disabled.

The input buffer shall contain 3 bytes

Byte #	Value	Description
B0	0x19	Escape command code
B1	0x00	LED number
B2	0x00	LED ON
	0x01	LED OFF

The output buffer is

Output buffer
NULL

6.7.6. READER_CNTLESS_GET_ATS_ATQB

This escape command enables the host to retrieve the ATS for Type A T= CL or the ATQB for Type B cards.

The input buffer contains the escape command code

Input buffer
0x93

The output buffer contains the ATS bytes or the ATQB bytes depending on the type of PICC placed on the reader.

6.7.7. READER_CNTLESS_GET_TYPE

This escape command retrieves the type of the card which SCL01X is configured to poll for.

The input buffer shall contain the escape command code

Input buffer
0x94

The output buffer shall point to a BYTE buffer which will contain the type value coded as

Value	Description
0x00	Type A
0x01	Type B
0x02	Type A + type B
0x18	FeliCa only
0x19	FeliCa + type A
0x1A	FeliCa + type B
0x1B	FeliCa + type A + type B

6.7.8. READER_CNTLESS_SET_TYPE

This escape command configures the type of cards SCL01X will poll for.

Using this command can improve the polling efficiency of SCL01X for applications where only type A or only type B cards are expected.

The default is Type A+type B+FeliCa (0x1B).

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x95	Escape command code
B1	0x00	Type A
	0x01	Type B
	0x02	Type A + type B
	0x18	FeliCa only
	0x19	FeliCa + type A
	0x1A	FeliCa + type B
	0x1B	FeliCa + type A + type B

The output buffer is

Output buffer
NULL

6.7.9. READER_CNTLESS_RF_SWITCH

This escape command can be used to retrieve/set the RF state of SCL01X.

The default RF field state is ON.

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x96	Escape command code
B1	0x00	Switch RF Field OFF
	0x01	Switch RF Field ON
	0xFF	Get current field state

After the RF is turned off, to turn the RF ON again, card connect shall be done in direct mode.

If B1 of the input buffer is 0x00 or 0x01 the output buffer is

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with 2 possible values

Output buffer	Description
0x01	RF field is OFF
0x00	RF field is ON

6.7.10. READER_CNTLESS_RAW_CFG

This escape command switches SCL01X to raw mode.

When SCL01X is in raw mode it only polls for one type of contactless card.

SCL01X is by default not in this mode and therefore READER_CNTLESS_RAW_XMIT would fail.

The input buffer contains 2 bytes

Byte #	Value	Description
B0	0x97	Escape Function code
B1	0x00	Type A will be use for further transmissions in raw mode
	0x01	Type B will be use for further transmissions in raw mode

The output buffer is

Output buffer
NULL

Once SCL01X is in raw mode commands can be sent using READER_CNTLESS_RAW_XMIT escape command.

6.7.11. READER_CNTLESS_RAW_XMIT

This escape command can only be executed by the firmware once SCL01X is put in raw mode using the READER_CNTLESS_RAW_CFG escape command.

This escape command can be used to send commands to smart card when SCL01X is in raw mode

The input buffer is

Byte #	Value	Description
B0	0x98	Escape Function code
B1		Wait Time Extension
B2		Is CRC specified?
B3		No of bits per command
B4		Command length
B5 - Bn		Command

The output buffer contains the response to the command.

The following example uses the raw mode to send a REQB command

First, we have to switch the SCL01X into raw mode for type B communication

Byte #	Value	Description
B0	0x97	READER_CNTLESS_RAW_CONFIG code
B1	0x01	Type B will be used

Then, we can send the following bytes to obtain the ATQB response of any type B user token in the field

Byte #	Value	Description
B0	0x98	READER_CNTLESS_RAW_XMIT code
B1	0x05	FWI is set to 5
B2	0x01	Enable CRC (CRC will be calculated by the RF front end of SCL01X)
B3	0x00	Number of bits to be sent in the command
B4	0x03	Command length in bytes
B5	0x05	REQB command's anticollision prefix byte
B6	0x00	REQB command's application family identifier
B7	0x01	REQB command parameter with slot number set as 1

```
ATR length: 13
ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68
APDU: FF CC 00 00 02 97 01
SW12: 9000 (OK)

APDU: FF CC 00 00 08 98 05 01 00 03 05 00 01
SW12: 9000 (OK)
DataOut: 50 51 63 FF FF 00 00 00 00 73 81 93 (12 byte(s))
```

6.7.12. READER_ CNTLESS_DISABLE_PPS

By default SCL01X does automatic PPS – i.e. it switches the RF communication speed to the highest possible supported by the card.

This escape command can be used to switch ON/OFF automatic PPS. When automatic PPS is OFF, then 106Kbps only is available

The input buffer is

Byte #	Value	Description
B0	0x99	Escape command code
B1	0x01	Disable Auto-PPS
	0x00	Enable Auto-PPS

The output buffer is

Output buffer
NULL

6.7.13. READER_CNTLESS_848KBPS

This escape command can be used to enable/disable 848kbps support by SCL01X as well as query whether 848kbps is currently enabled or disabled by SCL01X.

The RF communication with a user token will only switch to 848Kbps provided the user token supports this baudrate and provided automatic PPS is ON.

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x9D	Escape command code
B1	0x00	Disable 848Kbps support
	0x01	Enable 848Kbps support
	0xFF	Get current status on 848Kbps support

If B1 of the input buffer is 0x00 or 0x01 then the output buffer is

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with following possible values

Output buffer	Description
0x00	848Kbps is disabled
0x01	848Kbps is enabled

6.7.14. READER_CNTLESS_BAUDRATE

This escape command can be used to get the actual operating baud rate of card-reader communication.

The input buffer shall contain the escape message value.

Input buffer
0x9E

The output buffer shall point to a BYTE buffer with following possible values

Output buffer	Description
0x00	106Kbps in both directions
0x01	106Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x02	106Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x03	106Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x10	212Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x11	212Kbps in both directions
0x12	212Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x13	212Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x20	424Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x21	424Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x22	424Kbps in both directions
0x23	424Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x30	848Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x31	848Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x32	848Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x33	848Kbps in both directions

6.7.15. READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2

This escape command can be used to force baud rate for Contactless cards.

The input buffer is

Byte #	Value	Description
B0	0xAD	Escape command code
B1	0x00	Apply the baud rate specified by the card
	0x01	Force baud rate specified in B2
B2	b0- DR=2 supported, if bit is set to 1 b1- DR=4 supported, if bit is set to 1 b2- DR=8 supported, if bit is set to 1 b3- shall be set to 0, 1 is RFU b4- DS=2 supported, if bit is set to 1 b5- DS=4 supported, if bit is set to 1 b6- DS=8 supported, if bit is set to 1 b7- 1 if the same D is required for both communication directions b8- 0 if different D is supported for each communication direction	Encoding of the baud rate to be forced if B1 value is 0x01. No need to send this byte in case B1 has the value =x00
	NULL	If B1=0x00

The output buffer is

Output buffer
NULL

A card connect has to be done after this escape command is sent to switch baudrate to the value set by the user.

If user tries to force a higher baudrate on a Type A card which does not support higher baudrates, communication will occur at 106 kbps only. If the same was tried on a type B card then card connect will fail. To successfully connect to the card again the user has to connect at the baud rate specified by the card, by sending the escape command 0xAD00, and then do a 'card connect'.

6.7.16. READER_GETPNPPARAMS

This escape command is used to retrieve the VID, PID, serial number and PNP string.

The input buffer contains the escape command code

Input buffer
0xD3

The output buffer contents are described below.

Byte #	Description
B0-B1	VID(2 Bytes)
B2-B3	PID(2 Bytes)
B4-B17	Serial Number(14 Bytes)
B18	PNP String length (up to 127 bytes)
B19 - Bx	PNP String (up to 127 bytes)

6.7.17. READER_GETSLOTINFO

This escape command is used to retrieve the slot information.

The input buffer contains the escape command code

Input buffer
0xD9

The output buffer is

Byte #	Value	Description
B0	0x01	Only 1 slot supported on SCL01X
B1	0x00	Slot number
B2	0x01	Contactless slot

6.7.18. READER_GET_CARD_DETAILS

This escape command is used to get details about the PICC placed on the reader.

The input buffer contains the escape command code

Input buffer
0xDA

The output buffer is

Byte #	Value	Description
B0	0x00	Type A card
	0x01	Type B card
	0x04	FeliCa 212
	0x08	FeliCa 424
B1	0x00	Memory card
	0x01	TCL card
	0x02	Dual interface card
	0x43	FeliCa
B2		PUPI / UID Length
	0x08	For FeliCa cards
THEN EITHER		
B3-B12		PUPI/UID bytes 0x00 byte padding used if length smaller than 10
B13	0x00	CID not supported
	0x01	CID supported
B14	0x00	NAD not supported
	0x01	NAD supported
B15		Bit Rate Capability
B16		FWI
B17		IFSC
B18		MBLI
B19		SAK
B20		SFGI
OR		
B3–B10		8 Bytes NFCID2
B11		Request service command response time parameter (see JIS-6319 specification)
B12		Request response command response time parameter
B13		Authentication command response time parameter
B14		Read command response time parameter

B15		Write command response time parameter

Examples

MIFARE Ultralight	ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 00 07 04 E4 C3 D9 5B 02 80 00 00 00 00 00 00 04 0E 00 00 00 (21 byte(s))
MIFARE 1K	ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 00 04 FA 92 6C D6 00 00 00 00 00 00 00 00 00 04 0E 00 08 01 (21 byte(s))
MIFARE DESFire	ATR length: 6 ATR: 3B 81 80 01 80 80 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 01 07 04 7A 72 C9 92 26 80 00 00 00 01 00 77 08 3E 00 20 01 (21 byte(s))
Topaz	ATR length: 7 ATR: 3B 82 80 01 02 44 45 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 (1 byte(s))
ISO14443-4A	ATR length: 14 ATR: 3B 89 80 01 80 67 04 12 B0 03 02 01 00 49 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 01 04 08 66 41 4F 00 00 00 00 00 00 01 00 33 0C FE 00 20 04 (21 byte(s))
ISO14443-4B	ATR length: 13 ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 01 01 04 CA 9D FF FF 00 00 00 00 00 00 01 00 73 09 FE 00 00 00 (21 byte(s))
ISO14443-4A + MIFARE	ATR length: 19 ATR: 3B 8E 80 01 4E 58 50 53 6D 61 72 74 65 49 44 30 2E 31 57 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 00 02 04 00 89 0D 82 00 00 00 00 00 00 01 00 77 0C FE 00 38 00 (21 byte(s))
FeliCa	ATR length: 17 ATR: 3B 8C 80 01 04 43 FD 01 14 E4 00 7E 0A C0 31 93 50 APDU: FF CC 00 00 01 DA SW12: 9000 (OK) DataOut: 04 43 08 01 14 E4 00 7E 0A C0 31 4B 02 4F 49 93 (16 byte(s))

6.7.19. READER_IS_SCL01x

This escape message may be used to check if the connected device is an SCL01x device.

The input buffer contains the escape command code

Input buffer
0xDB

The output buffer shall point to a BYTE buffer with following possible values

Output buffer	Description
0x01	SCL01x device connected
NULL	Device connected is not SCL01x

6.7.20. READER_SEND_ATTRIB_WITH_INF

This Escape command may be used to send application layer bytes to a type B card along with the ATTRIB command in the higher layer INF field.

The input buffer is

Byte #	Value	Description
B0	0xE2	Escape command code
B1 onwards		INF Bytes

6.7.21. USER_TRACEABILITY_DATA

6.7.21.1. Description

A new feature is added in SCL01x firmware Ver 1.12 and above where a user can store & retrieve data from the reader. This data is stored in the chip and hence data is retained even after un-plug and re-plug.

Note : Please be informed that very frequent write should be avoided (E² supports only 100K writing)

Escape IOCTL (0xF0) can be used to read and write to this user configuration area. Max of 249 bytes can be read and written and the sector has to be read and written as a whole.

6.7.21.2. Format

Byte #	Value	Description
B0	0xF0	Escape command code
B1	0x01	Read the 249 bytes.
	0x02	Write 249 bytes inside the buffer.

To read the contents set the InputBuffer of the escape IOCTL to “0xF0 0x01” and send the escape IOCTL. Response will be 249 bytes of data.

To write data set the InputBuffer of the escape IOCTL to “0xF0 0x02 {249 bytes data}”. No response bytes are sent for this IOCTL. The return value of ScardControl denotes whether the write has passed or not.

Note : If complete data (249 bytes) is not given during write operation then random data will be padded to the given data and then written. If you want to modify only partial data in the reader read the entire 249 bytes modify the data you want and then write it back to the reader.

6.7.22. READER_DISABLE_ENABLE_POLLING

6.7.22.1. Description

This escape command can be used to enable/disable card polling by reader once a user token has been detected.

This IOCTL is primarily used to stop polling before starting an authentication session and after completing the session. E.g Mifare Plus AES or sector authentication, Ultralight C AES authentication etc.

IMPORTANT NOTE :

Once the user token is detected, once the polling is disabled, the reader will not be able to detect any token removal and the application will still display as card present. We need to enable the polling again in order to detect card removal.

Reader will not be able to detect new cards before the polling is enabled again.

This command must be used with care, and application developers must ensure that polling is enable again as soon as it is not needed to have the polling disabled.

6.7.22.2. Format

Input buffer

Byte #	Value	Description
B0	0xAC	Escape command code
B1	0x00	Enable Polling
	0x01	Disable Polling
	0xFF	Gets Current state of polling.

Output buffer

Byte #	Configuration parameter value from input buffer	Output buffer
NA	0x00 or 0x01	NULL
B0	0xFF	0x00 polling enabled
		0x01 polling disabled

7. Annexes

7.1. Annex A – Status words table

SW1	SW2	Description
0x90	0x00	NO ERROR
0x67	0x00	LENGTH INCORRECT
0x6D	0x00	INVALID INSTRUCTION BYTE
0x6E	0x00	CLASS NOT SUPPORTED
0x6F	0x00	UNKNOWN COMMAND
0x63	0x00	AUTHENTICATION ERROR
0x65	0x81	STATUS_COMMAND_FAILED
0x65	0x91	STATUS_SECUIRTY_STATUS_NOT_MET
0x68	0x00	CLASS BYTE INCORRECT
0x6A	0x81	FUNCTION NOT SUPPORTED
0x6B	0x00	WRONG PARAMETER P1-P2

7.2. Annex B – Sample code using escape commands through Escape IOCTL

```
File Name :  T_pupi.H

#ifdef __cplusplus
extern "C" {
#endif


#define IOCTL_CCID_ESCAPE                           SCARD_CTL_CODE (0xDAC)

#define CCID_GET_PUPI_STATUS                0xFF9B
#define CCID_SET_PUPI_ON                            0x019B
#define CCID_SET_PUPI_OFF                           0x009B

#define MINTIMEOUT                                  300

#ifdef __cplusplus
}
#endif

File Name :  T_pupi.CPP


#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "T_pupi.H"


VOID main(VOID)
{


        SCARDCONTEXT            ContextHandle;
        SCARDHANDLE             CardHandle;
        BYTE                            OutByte;
        WORD                            InWord,i;
        DWORD                   ActiveProtocol;         /* ICC protocol */
        ULONG                           InBufLen,ResLen;
        ULONG                           ret;
```

```
        SCARD_READERSTATE      Reader[1];



// please add the name of the used reader here or use SCardListReaders
// to find the right reader name
        char    *ReaderName[] = {"SCM Microsystems Inc. SCL01X USB ContactlessReader 0",
                                 NULL};

/*****************************************************************************
*****************/

        ContextHandle = -1;

        ret = SCardEstablishContext(SCARD_SCOPE_USER, NULL, NULL, &ContextHandle);

        if (ret == SCARD_S_SUCCESS)
        {
                ret = SCardConnect(    ContextHandle,
                                       ReaderName[0],
                                       SCARD_SHARE_SHARED,
                                       SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                                       &CardHandle,
                                       &ActiveProtocol);

                if (ret == SCARD_S_SUCCESS)
                {
                        /* get actual PUPI status: ON/OFF */

                        InBufLen = 2;
                        InWord = CCID_GET_PUPI_STATUS;
                        ret = SCardControl (CardHandle,
      IOCTL_CCID_ESCAPE,
                                            &InWord,
                    InBufLen,
                                            &OutByte,
              1,
      &ResLen);

                        printf ("\n Get PUPI status: %lx: %.2x", ret,OutByte);

                        Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
                        Reader[0].dwEventState = SCARD_STATE_UNAWARE;
                        Reader[0].szReader = ReaderName[0];
```

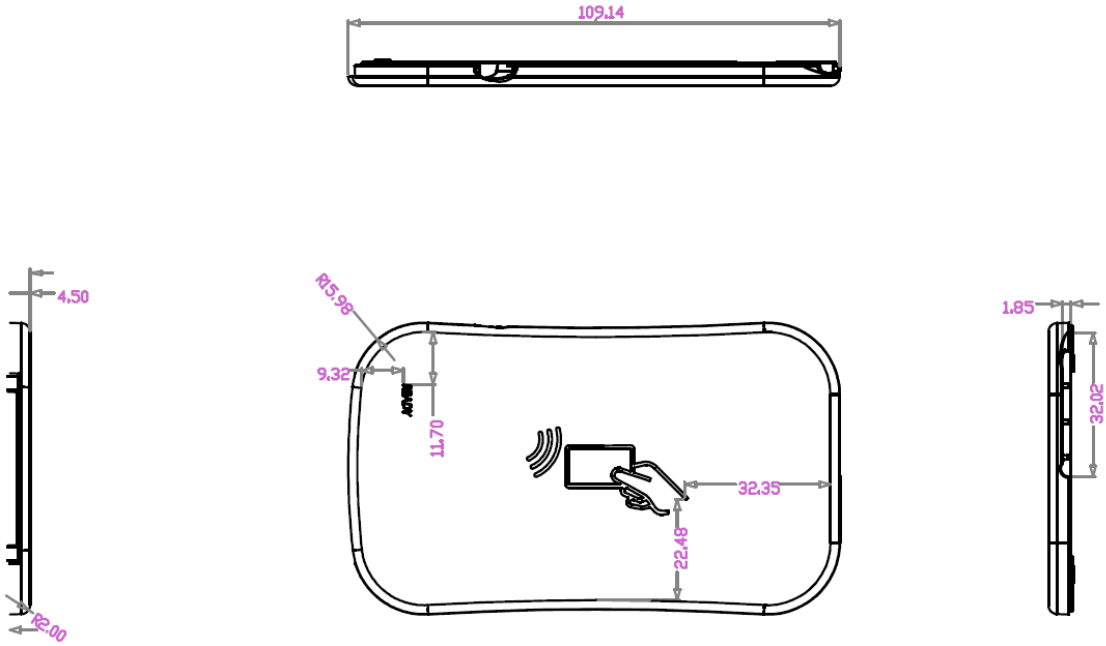
```
                        ret = SCardGetStatusChange( ContextHandle, MINTIMEOUT, Reader,
1);

                        printf ("\nATR: ");
                        for (i=0; i<Reader->cbAtr; i++)
                        {
                                printf ("%.2x ",Reader->rgbAtr[i]);
                        }
                        printf ("\n-----------------------------------------\n");


                        /* set actual PUPI status: ON */

                        printf ("\nenable PUPI ");
                        InBufLen = 2;
                        InWord = CCID_SET_PUPI_ON;
                        ret = SCardControl (CardHandle,
                                                        IOCTL_CCID_ESCAPE,
                                        &InWord,
                InBufLen,
                                        &OutByte,
                1,
         &ResLen);

                        ret = SCardDisconnect(CardHandle, SCARD_RESET_CARD);

                        ret = SCardConnect (ContextHandle,
                                        ReaderName[0],
                                        SCARD_SHARE_SHARED,
                                        SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                                        &CardHandle,
                                        &ActiveProtocol);

                        /* get actual PUPI status: ON/OFF */

                        InBufLen = 2;
                        InWord = CCID_GET_PUPI_STATUS;

                        ret = SCardControl (CardHandle,
        IOCTL_CCID_ESCAPE,
                                        &InWord,
        InBufLen,
                                        &OutByte,
                1,
        &ResLen);
```

```
                printf ("\n Get PUPI status: %lx: %.2x", ret,OutByte);

                Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
                Reader[0].dwEventState = SCARD_STATE_UNAWARE;
                Reader[0].szReader = ReaderName[0];

                ret = SCardGetStatusChange (ContextHandle,
        MINTIMEOUT,
Reader, 1);
                printf ("\nATR: ");
                for (i=0; i<Reader->cbAtr; i++)
                {
                        printf ("%.2x ",Reader->rgbAtr[i]);
                }
                printf ("\n----------------------------------------------\n");



                /* set actual PUPI status: OFF */
                printf ("\ndisable PUPI ");
                InBufLen = 2;
                InWord = CCID_SET_PUPI_OFF;
                ret = SCardControl(CardHandle, IOCTL_CCID_ESCAPE,
                                        &InWord, InBufLen,
                                        &OutByte, 1, &ResLen);

                ret = SCardDisconnect(CardHandle, SCARD_RESET_CARD);
                ret = SCardConnect(ContextHandle,
                                       ReaderName[0],
                                       SCARD_SHARE_SHARED,
                                       SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                                       &CardHandle,
                                       &ActiveProtocol);


                /* get actual PUPI status: ON/OFF */
                InBufLen = 2;
                InWord = CCID_GET_PUPI_STATUS;
                ret = SCardControl(CardHandle, IOCTL_CCID_ESCAPE,
                                        &InWord, InBufLen,
                                        &OutByte, 1, &ResLen);
                printf ("\n Get PUPI status: %lx: %.2x", ret,OutByte);

                Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
                Reader[0].dwEventState = SCARD_STATE_UNAWARE;
```

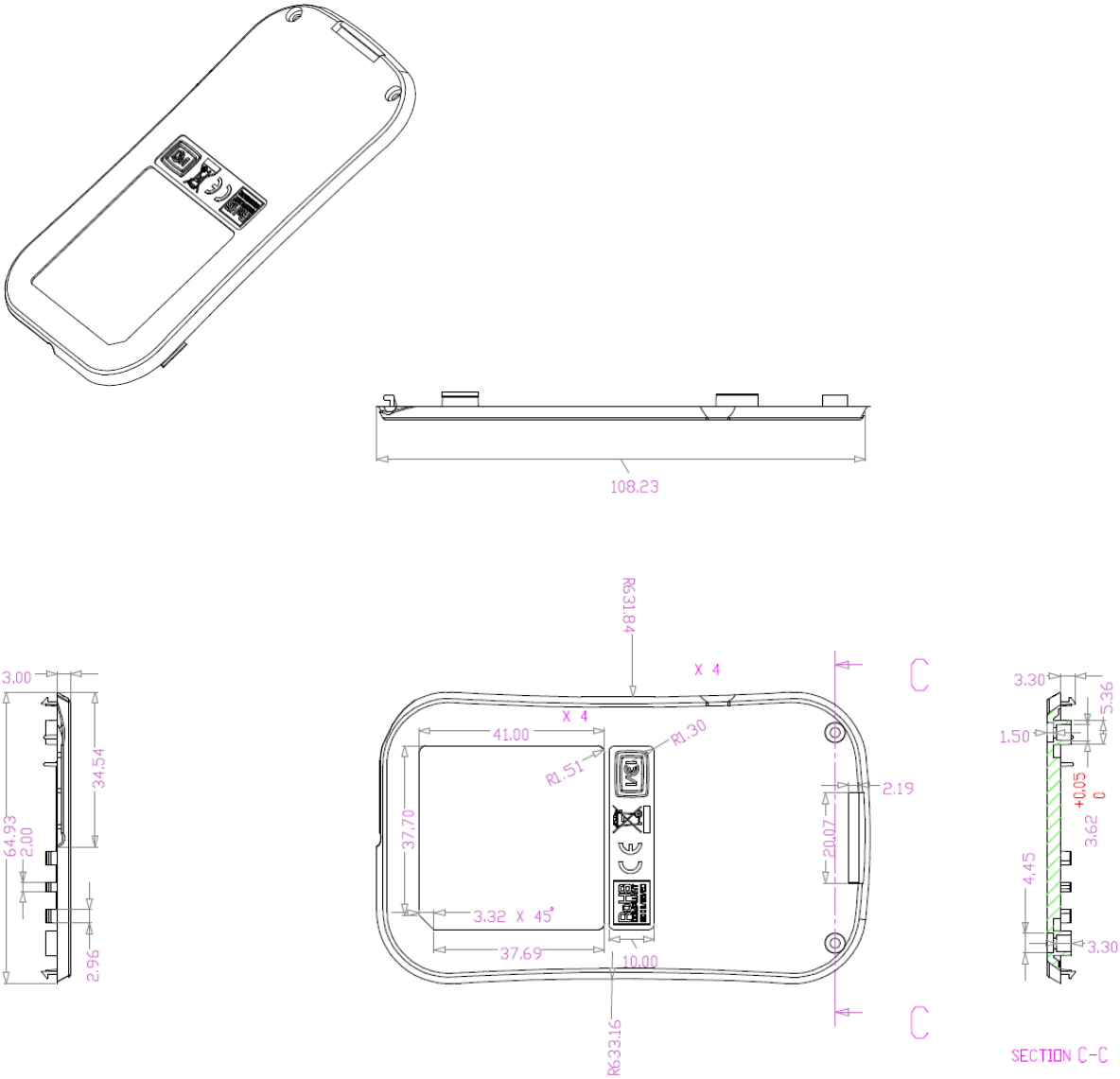
```
                        Reader[0].szReader = ReaderName[0];
                        ret = SCardGetStatusChange( ContextHandle, MINTIMEOUT, Reader,
1);
                        printf ("\nATR: ");
                        for (i=0; i<Reader->cbAtr; i++)
                        {
                                printf ("%.2x ",Reader->rgbAtr[i]);
                        }
                        printf ("\n-----------------------------------------------\n");

                        ret = SCardDisconnect(CardHandle, SCARD_RESET_CARD);
                }
                else
                {
                        printf("\n SCardConnect failed with 0x%.8lX",ret);
                }
                ret = SCardReleaseContext(ContextHandle);
        }
        else
        {
                printf("\n SCardEstablishContext failed with %.8lX",ret);
        }

        printf("\npress any key to close the test tool\n");
        getch();
}
```

7.3. Annex C – Mechanical drawings

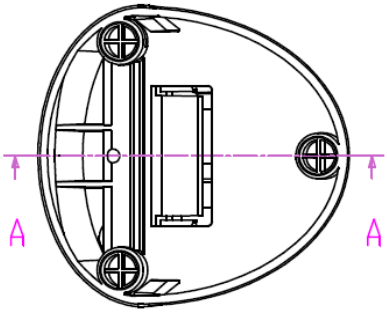
7.3.1. Top casing

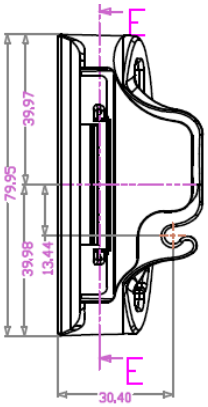


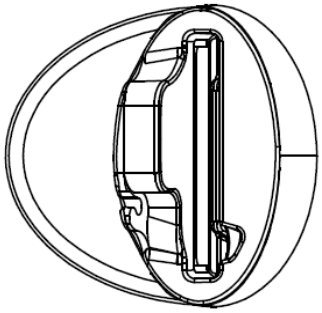
7.3.2. Bottom casing

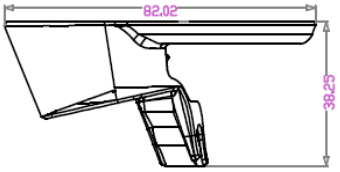


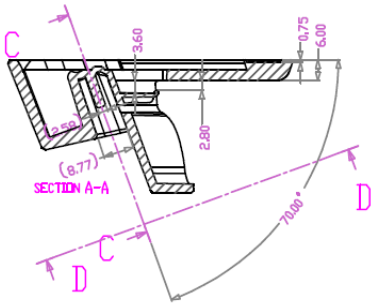
7.3.3. Cradle

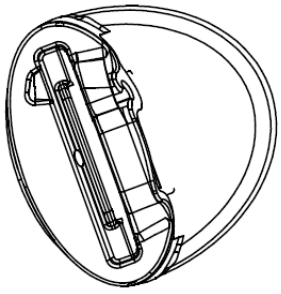

A
A

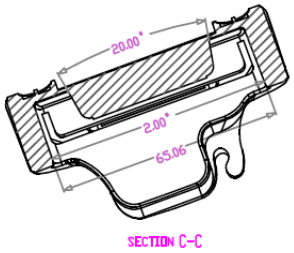

E
39,97
79,95
39,98
13,44
E
30,40




82,02
38,25


C
3,60
0,75
6,00
2,80
(8,77)
SECTION A-A
D
C
D
70,00°


SECTION D-D


20,00°
2,00°
65,06
SECTION C-C

7.3.4. Snap-on card holder